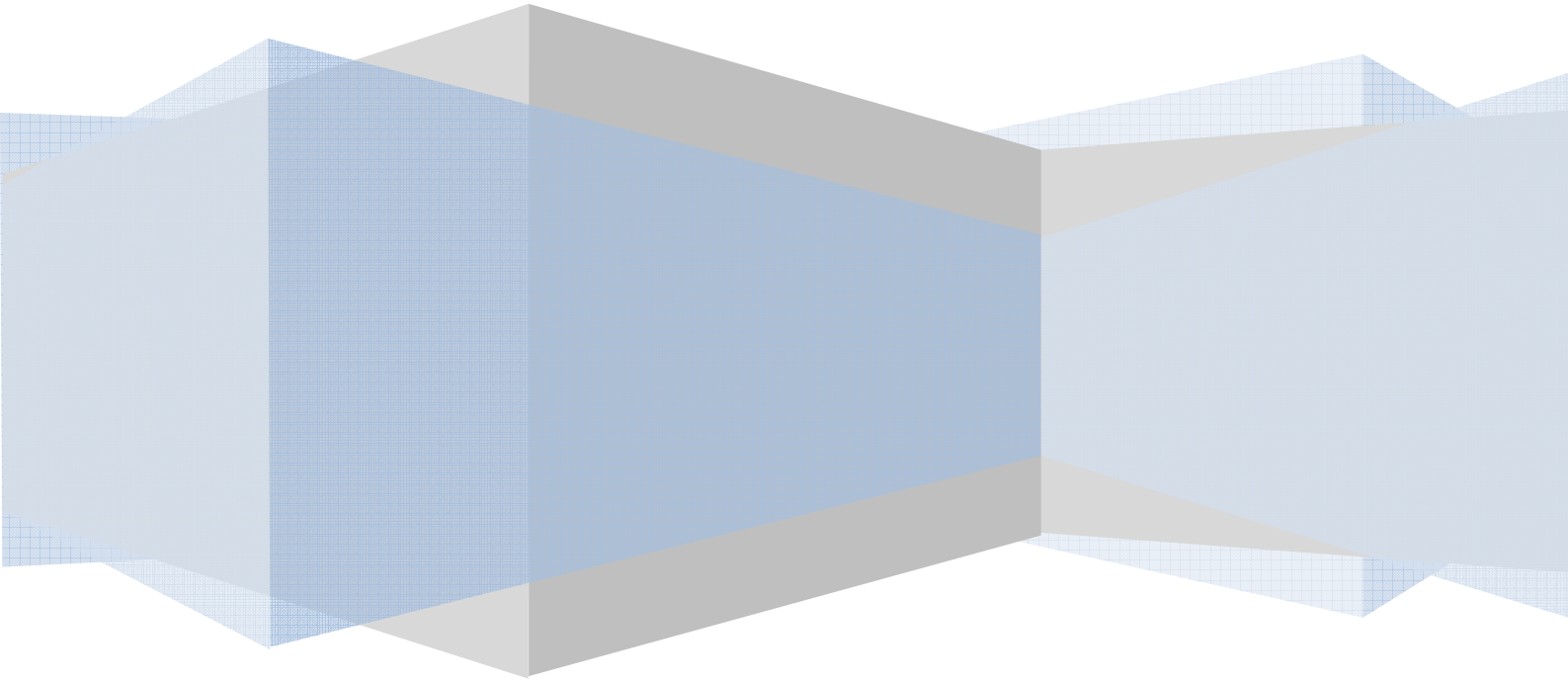
Aplicación de cuestionarios para Android

PROYECTO FIN DE CARRERA

Miguel Degayón Cortes

Tutor: Alejandro Calderón Mateos



1 ÍNDICE

2 ÍNDICE DE ILUSTRACIONES

3 ÍNDICE DE TABLAS

4 INTRODUCCIÓN

Este capítulo pretende ser el punto de entrada al resto del documento, ofreciendo una visión general sobre el tema que nos ocupa y pudiendo encontrar en él los objetivos y motivaciones que han llevado a realizar el presente proyecto además de una breve descripción del resto de contenidos.

4.1 Visión general

En el marco de la universidad Carlos III, es frecuente disponer de herramientas web que permitan el acceso o intercambio de recursos docentes entre profesores y alumnos. Sirva como ejemplo el portal Aula Global: desde esta herramienta web de la universidad, el alumno puede disponer de recursos (lecciones, apuntes, lecturas recomendadas), tareas (ejercicios y prácticas que puedes realizar y entregar directamente en la aplicación) o cuestionarios.

Este proyecto nace con la intención de llevar estas ideas al ámbito de la telefonía móvil, aprovechando las posibilidades que ofrece la actual generación de teléfonos inteligentes (o smartphones, como prefiera), para convertir nuestro dispositivo móvil en una herramienta de ayuda a la docencia. De forma más específica, **la aplicación desarrollada permitirá realizar tests de auto-evaluación,** haciendo posible que el alumno pueda hacer un seguimiento de sus progresos.

El software desarrollado, que ha recibido el nombre de testing, ha sido concebido como la parte cliente de un sistema Cliente-Servidor. La aplicación móvil requerirá del usuario/alumno completar un proceso de identificación. Tras lo cual, se descargará al teléfono (o actualizará) la colección de cuestionarios disponible para el alumno.

De esta manera, y al igual que sucede en la herramienta antes aludida, Aula Global, cada usuario tendrá acceso a los test de las asignaturas por él cursadas.

La forma en que el alumno recibe acceso al sistema para lograr identificarse, es algo que va más allá de este proyecto. (Una solución válida, y habitual en la universidad, sería la de facilitar, vía correo electrónico, el nombre de usuario y la contraseña necesaria para identificarse en un sistema). De igual forma, y como se mencionó en el párrafo inmediatamente anterior, es del cliente del que nos ocuparemos en este proyecto, es por esto que no se llevará a cabo un desarrollo completo del servidor. Sin embargo, se elaborará una pequeña aplicación, que hará las veces de servidor. Su funcionalidad se va a limitar a implementar el protocolo de red creado de forma expresa para la descarga de los

cuestionarios, sin que sus responsabilidades vayan más allá de esto.

4.2 Objetivos

El presente proyecto tiene, como su objetivo principal, el de desarrollar una aplicación móvil para auto-evaluar los conocimientos de los alumnos mediante cuestionarios. Más allá de esta meta, el resto de propósitos del proyecto se enumeran a continuación:

- Ofrecer una panorámica de las principales plataformas móviles que existen en el mercado en la actualidad, analizando sus principales características.

- Dado que será Android el sistema operativo sobre el que se ejecutará la aplicación, se tratará en detalle la arquitectura del S.O., enumerando y analizando sus principales componentes.

- Recoger los métodos y técnicas de ingeniería del software empleados durante cada una de las etapas del desarrollo de la aplicación. Además, se destacarán algunas de las decisiones de diseño y detalles relativos a la implementación. Principalmente, aquellos que tienen que ver con las particularidades de trabajar con dispositivos móviles.

4.3 Motivación

Además de los objetivos del proyecto, sería conveniente hablar de qué motiva la creación de la aplicación y lo que con ella se pretende conseguir. TESTING surge a partir de la idea de que el alumno pueda, desde un dispositivo móvil y en cualquier momento, evaluar sus conocimientos sobre las asignaturas que cursa. A mis ojos, eran numerosos los escenarios en los que una aplicación así sería de utilidad para el alumno. Me veía a mi mismo en una sala de espera o en un vagón de metro de vuelta a casa desde el campus, jugueteando con el móvil: leyendo noticias, jugando algún juego o revisando mi perfil en una red social. ¿No sería genial poder emplear mi teléfono en esos momentos para algo más que llenar el tiempo? Mejor aún, piensa en el alumno que viaja en transporte público a la universidad para hacer un examen. ¿No querría él ponerse a prueba en estos últimos momentos, para asegurarse de que está realmente preparado?

Desde Aula Global, los alumnos de la universidad Carlos III ya disponen de la posibilidad de realizar test de auto-evaluación, por lo que podrían acceder a estos cuestionarios desde cualquier teléfono con conexión a Internet. El lector podría creer que esto, si no convierte a TESTING en una aplicación inútil, sí que hace más difícil justificar su desarrollo. Voy a explicar porque la idea de TESTING, incluso en esta situación, no sólo sigue siendo válida, sino que se trata de la opción más adecuada.

El principal inconveniente de Aula Global es que no está ideada para mostrarse en un teléfono móvil. A menudo, existen problemas de conectividad que provocan la pérdida de la

conexión a Internet del móvil. El teléfono viaja con nosotros y la cobertura de las redes depende de nuestra ubicación. Con frecuencia, en el interior de un edificio, en un vagón de Metro o cuando el vehículo en que viajamos atraviesa un túnel, la conexión a la red móvil del teléfono se pierde. El intercambio de información entre el navegador y el servidor web que aloja Aula Global es lo suficientemente frecuente como para que estas pérdidas de conexión le afecten.

Además, la web de Aula Global no está adaptada a la interacción con un teléfono: la sencilla tarea de hacer clic en una de las asignaturas del portal se complica al sustituir el monitor de nuestro ordenador por la pantalla de un móvil y al reemplazar el ratón por el dedo índice de nuestra mano.

Con TESTING, el problema de la conectividad se minimiza, pues únicamente hace uso de la conexión a la red móvil al inicio: tras completar el proceso de login, se descargará del servidor los nuevos tests disponibles para el usuario, de existir. Además, la aplicación está concebida desde el comienzo para mostrarse en la pantalla del teléfono, con la ventaja que esto conlleva en la interacción del usuario con el programa. Todo lo anterior hace que, no sólo sea compatible la existencia de TESTING con la aplicación de Aula global, sino que, en el caso de acceder desde un teléfono móvil, sea preferible a la alternativa Web.

4.4 Etapas del desarrollo

El primer paso fue el de llevar a cabo una labor de documentación. En primer lugar, sobre Android, algo lógico cuando uno encara el desarrollo sobre una plataforma desconocida. Se recopiló información sobre la arquitectura y aspectos sobre la programación de aplicaciones (lenguaje, SDKs, interfaz de usuario).

Siguió la labor investigadora analizando aplicaciones para el sistema operativo, a fin de adquirir una mayor comprensión de las posibilidades del sistema, buenas prácticas en la creación de interfaces de usuarios y aspectos más secundarios.

El siguiente paso, necesario, era la definición de la aplicación a desarrollar. Y tras ello, comparar aplicaciones de funcionalidad parecida, tomando nota de sus aciertos y deficiencias, tratando de evitar éstas en el ulterior proceso de creación del software.

Una vez elegida la aplicación, llegó el momento del desarrollo de la aplicación. Es ahora cuando tengo la ocasión de poner en marcha los conocimientos adquiridos en la universidad, pues tengo la oportunidad, con la creación de Testing, de llevar a la práctica mi saber en el ámbito de la ingeniería del software. Y en relación a lo anterior, tiene su origen la redacción de este documento –al menos, en una parte: Pues ha de demostrar mi recién mencionada capacidad para completar un proceso de desarrollo mediante las técnicas de ingeniería.

Con las pruebas sobre el software se puso fin al desarrollo de la aplicación, corrigendo errores encontrados y mejorando los aspectos menos acertados encontrados y optimizando el funcionamiento de la misma.

4.5 Resto del documento

El objetivo de este apartado es el de ofrecer una breve descripción del contenido de cada uno de los siguientes capítulos y anexos que componen el resto del documento.

En el segundo capítulo, "El estado de la cuestión", repasaremos los principales Sistemas operativos móviles, enumerando sus puntos fuertes y sus desventajas. Termina este capítulo con la descripción de aplicaciones de funcionalidad similar a TESTING.

Durante el tercer capítulo, "Android", enumeraremos las más importantes características del Sistema operativo (diseño y arquitectura), además de hacer un repaso por su historia y desde su primera versión.

EL siguiente capítulo "Análisis, diseño e implementación de la aplicación TESTING" incluye las principales etapas e hitos del desarrollo del software: su análisis, diseño e implementación.

Con el fin de detallar el presupuesto del proyecto y los costes asociados al mismo, se incluye el capítulo "Planificación y Presupuesto". En él, además se incluye la planificación del proyecto, que detalla las tareas que lo componen y el tiempo necesario estimado para completarlas. . Además de los recursos humanos necesarios, se incluye un informe en detalle de los equipos hardware empleados.

En el capítulo "Conclusiones y trabajos futuros" se dan a conocer las conclusiones finales alcanzadas así como los resultados obtenidos. También se exponen las posibles líneas de trabajo que se desprenden del desarrollo de esta aplicación y su utilidad.

El final del documento lo integran dos anexos que complementan la memoria: un glosario (que incluye términos y acrónimos) y un segundo de referencias, bibliografía y documentación empleada.

5 ESTADO DE LA CUESTIÓN

Se dedicará este capítulo a situar el proyecto dentro de un ámbitomás amplio en las tecnologías móviles, además de repasar varias aplicaciones de temática similar a TESTING disponibles en el Market.

5.1 Sistemas operativos para dispositivos móviles

Para la realización del presente proyecto, se ha utilizado el sistema operativo Android cuyas características serán convenientemente explicadas en el siguiente capítulo. En este apartado hablaremos del resto de alternativas existentes en el mercado que fueron valoradas antes realizar una elección.

A continuación se muestra una tabla comparativa de los sistemas operativos para dispositivos móviles analizados en esta sección:

	Windows Phone 7	IOS 4.0	Symbian	BlackBerry OS 4.7	Android
Requerimientos mínimos del sistema	SI	SI	No	NO	NO
Tipo de pantalla	Capacitiva	Capacitiva	Capacitiva/ Resistiva	Capacitiva	Capacitiva/ Resistiva
¿Fabricante único?	No	Si	No	Si	No
Tipo de Sistema Operativo	Cerrado	Cerrado	Abierto	Cerrado	Abierto
Soporte de memoria externa	Si	No	Si	Si	Si
Multitarea	No	SI	Si	Si	Si
Multitáctil	Si	SI	Si	Si	Si
Fragmentación	No	No	No	No	Si
Tienda de aplicaciones	MarketPlace	App Store	Ovi Store	App World	Market
MMS	Si	SI	Si	Si	Si
Soporte Adobe Flash	No	No	Si	No	Si
Tethering	No	SI	Si	Si	Si
Interfaz personalizable	Si	No	Si	Si	Si

5.1.1 Symbian

Propiedad de Nokia [5], se trata del sistema operativo móvil más extendido en la actualidad. Si bien el mejor momento de este S.O. parece haber pasado.

La actual versión SYMBIAN^3 fue pensada para correr sobre smartphones, incluyendo entre sus novedades: soporte HDMI, arquitectura en gráficos 2D y 3D, y mejoras en la interfaz de usuario, entre ellas la consistencia, solo menú de opciones y e incluso escritorios.

Sin embargo, y pese a los esfuerzos de Nokia por poner al día su sistema operativo, parece haberse quedado atrás respecto a sus competidores. Temo que, en cuanto a funcionalidades disponibles, Symbian [9] no está a la altura de lo exigido, por lo que ha de rechazarse el S.O. como plataforma para el proyecto.

5.1.2 iOS

Se trata del sistema operativo propiedad de Apple[1] que incorporan algunos de sus más famosos productos (IPad, IPhone o IPod Touch). Actualmente en su versión 4.3, este sistema se ha distinguido por ofrecer una experiencia de usuario intuitiva y agradable, además de responder de forma muy fluida. En europa, la cuota de mercado de IOS cuadriplica a la de Android, por lo que desarrollar para este sistema operativo podría parecer mejor opción.

Sin embargo, existen ciertos inconvenientes que desaconsejan su elección como plataforma para el presente desarrollo: Si bien el SDK es gratuito, sólo está disponible para MAC OS X - el sistema operativo de Apple para ordenadores de sobremesa y portátiles. Además, los costes de distribuir una aplicación oscilan entre 99 dólares (tasa a pagar por subir una aplicación a la AppStore, la tienda de aplicaciones para IOS) y 299 dólares (tasa anual del paquete de empresas).

Lo anteriormente mencionado descarta IOS [4] como la plataforma sobre la que desarrollar el proyecto.

5.1.3 BlackBerry OS

Es el sistema operativo móvil desarrollado por Research In Motion (RIM) para sus dispositivos BlackBerry. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM [7] para su uso en computadoras de mano, particularmente la trackwheel, trackball, touchpad y pantallas táctiles.

Estos dispositivos permiten el acceso a correo electrónico, navegación web y sincronización con programas como Microsoft Exchange o Lotus Notes aparte de poder hacer las funciones usuales de un teléfono móvil.

Son características más que apetecibles para ser el sistema operativo escogido para el proyecto. No obstante, El SO BlackBerry está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. El ámbito académico del proyecto, desaconseja su elección.

5.1.4 Windows Phone 7 (WP7)

El nuevo sistema desarrollado por Microsoft que sustituyó a Windows Mobile. WP7 [10] supone una renovación con respecto al anterior S.O. móvil de Microsoft, rompiendo totalmente con este presentado una nueva interfaz.

Una de las principales características en esta nueva interfaz es la pantalla principal, presentada en lo que Microsoft ha llamado mosaicos dinámicos que se actualizan constantemente en función de las aplicaciones que el usuario utiliza. Otro de los nuevos conceptos son los HUB, que agrupan funcionalidades y aplicaciones referentes a un mismo tema mejorando así la usabilidad del sistema.

WP7 aun está dando sus primeros pasos y la primera gran actualización del S.O. todavía no ha visto la luz. El poco tiempo pasado desde su lanzamiento hace que no goce todavía de la aceptación de los usuarios, con una cuota de mercado muy pequeña, lo que lo descarta como el SO sobre el que desarrollar el proyecto.

5.2 Aplicaciones móviles: tests y cuestionarios.

En la tienda de aplicaciones de Android podemos encontrar multitud de ejemplos de desarrollos similares. Son numerosas las aplicaciones dedicadas a ofrecer cuestionarios en forma de test a los usuarios.

La temática es muy diversa: test sobre conducción, historia de EE.UU., ciudades del mundo. Se escogerá una pequeña colección y, sobre ella, se analizarán sus características, apuntando sus aciertos y deficiencias, a fin de solucionar estos últimos y tener en cuenta los primeros de cara al desarrollo de TESTING.

5.2.1 AutoescuelaFacil.com

Se trata de una aplicación para resolver test de conducir, con el objetivo de preparar el examen necesario para adquirir diferentes permisos (turismos, motocicletas, ciclomotores…).Posee una amplia colección de test (en la versión gratuita analizada su número es más reducido) bien clasificados por categorías (tipo de permisos, test temáticos y de examen). Todas las preguntas del test se muestran en la misma pantalla, haciendo uso del scroll vertical para navegar entre ellas. Al corregir el test, muestra de forma bastante clara las preguntas falladas y acertadas. Todas las preguntas del test se corrigen a la vez, al tocar el botón correspondiente.

Ilustración 1: Pantalla de inicio de AutoescuelaFacil

Entre sus defectos señalaré el abuso del scroll vertical (al navegar por los menús, al realizar el test o al corregirlo), botones que no muestran un aspecto 'clicables' y una forma poco adecuada de presentar la corrección del test (es necesario recorrer todas las preguntas del test, mediante scroll vertical, para comprobar aciertos y fallos) y cierta falta de opciones (cuenta atrás, repetición del test…)

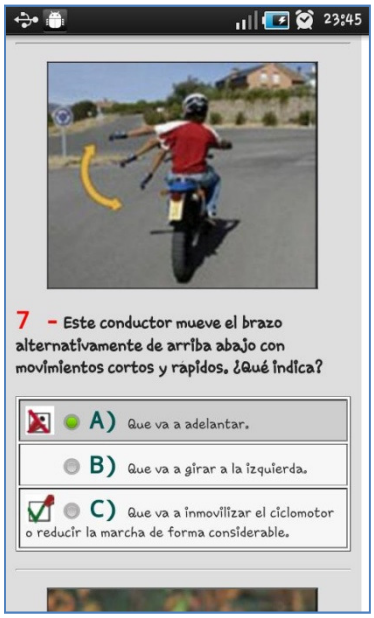


Ilustración 2: Pregunta de test de AutoescuelaFacil

5.2.2 CET 2009

Se trata de una aplicación con tests sobre física, química, matemáticas y biología. Sirve para preparar el examen CET.

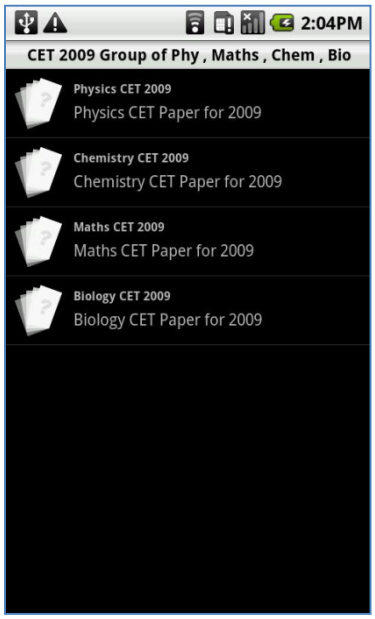


Ilustración 3: Inicio de la aplicación CET2009

Posee una menor colección de tests, clasificados en función de la temática. Todas las preguntas del test se corrigen a la vez, al seleccionar la opción 'Submit' que se muestra al hacer clic en el botón de menú del teléfono. Se muestra una pregunta por pantalla, y es un acierto la forma en que se presenta el resultado del test: Dando en primer lugar el número de aciertos y fallos y, mediante un clic en la opción deseada, navegar entre las anteriores. Se incluye un límite de tiempo – que no es configurable- para completar el test.

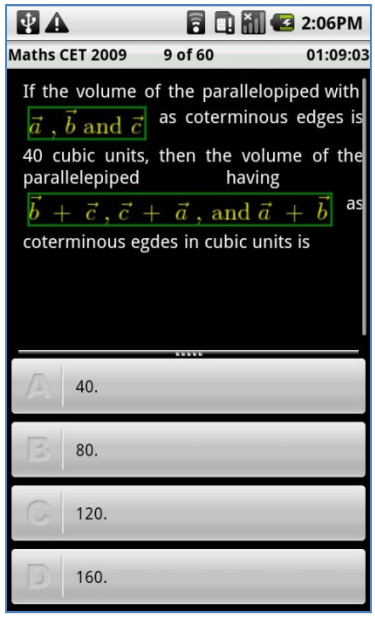


Ilustración 4: Pregunta de la aplicación CET2009

Su principal defecto es, a mi modo de ver, la falta de visibilidad de alguno de sus controles. Para navegar entre preguntas es necesario desplazarse por la pantalla arrastrando el dedo de derecha a izquierda. Esto, no es evidente ni es explicado en la aplicación. Se permite ocultar las opciones de respuesta para evitar el scroll en el enunciado, sin embargo la visibilidad del botón no es adecuada. En ocasiones el botón 'ATRÁS' del teléfono no responderá como se espera. Encuentro equivocado el uso del botón 'MENU' del teléfono para acciones elementales de la aplicación (corregir el test o mostrar los resultados tras completarlo).

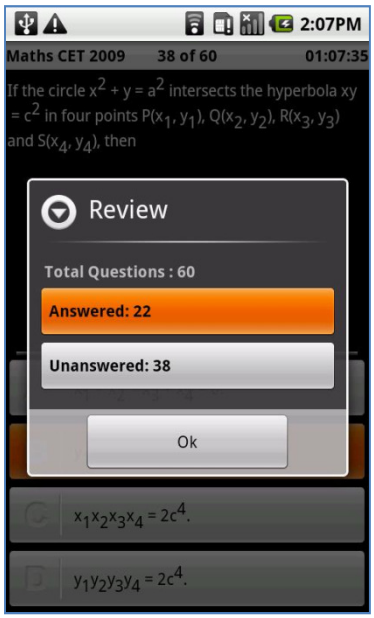


Ilustración 5: Fin de un test de CET2009

5.2.3 Countries of the world

Esta aplicación permite aprender las capitales, poblaciones, idiomas, PIB, las religiones, banderas y más de un gran número de países del mundo. Además posee el modo Quiz, que permite poner a prueba los conocimientos del usuario a este respecto. Este modo es el que vamos a analizar.

El modo de tests incluye novedades interesantes respecto a las anteriores: Permite incluir un reloj – de forma opcional – para controlar el tiempo empleado en el test. Cada pregunta se corrige de manera individual, adecuado para mostrar el resultado. Además se indica el porcentaje de respuestas acertadas en todo momento.

Ilustración 6: Pregunta de test de la aplicación Countries of the world

Entre sus defectos se encuentra el tamaño de las respuestas: En pantallas pequeñas hace que sea fácil cometer un error al seleccionarla. Algunas imágenes poseen un aspecto que las hace parecer 'clicables' sin serlo.

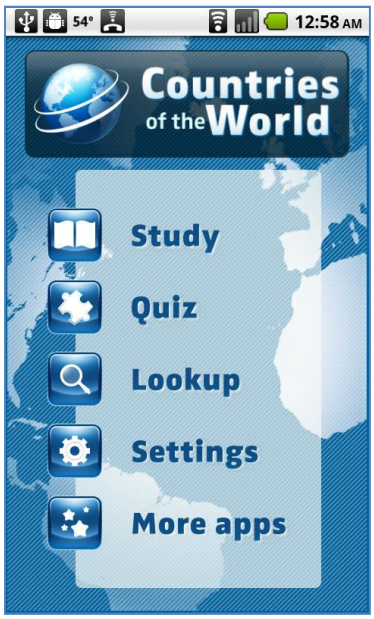


Ilustración 7: Pantalla de inicio de Countries of the world

5.2.4 Tests de conducir

Se trata de otra aplicación de test de auto-escuela. Es bastante similar al anterior, aunque corrige algunos de los fallos de 'auto-escuela fácil' y comete otros nuevos.

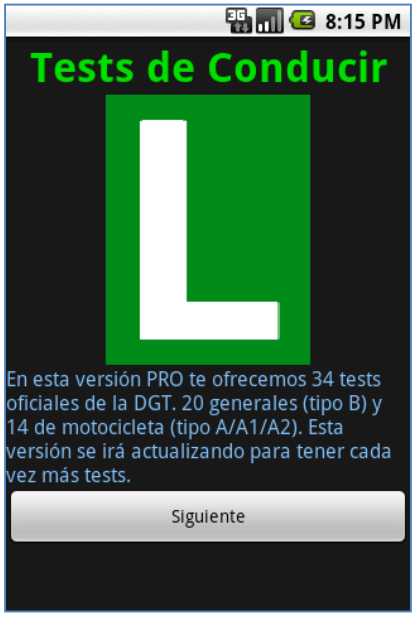


Ilustración 8: Pantalla de inicio de Tests de Conducir

En esta ocasión se ha solucionado el problema de visibilidad de los controles. Por el contrario, han reemplazado los radio-botones por check-boxes de forma extraña y la forma en que se muestra la corrección del test es insuficiente, pues sólo se incluye el número de la pregunta fallada, sin ser posible conocer la opción correcta.

Ilustración 9: Pregunta de la aplicación tests de conducir

5.2.5 Math Practice

Esta aplicación permite realizar cuestionarios sobre operaciones aritméticas básicas (suma, resta, multiplicación, división) y posee algunas características reseñables.

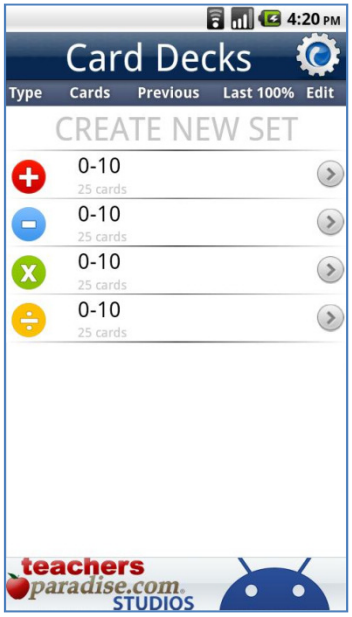


Ilustración 10: Inicio de Math Practice

Los cuestionarios están clasificados por el tipo de operación. Cada pregunta se corrige en el momento de la respuesta, como en 'Countries of the world', aunque en este caso, el tamaño de los botones es adecuado.

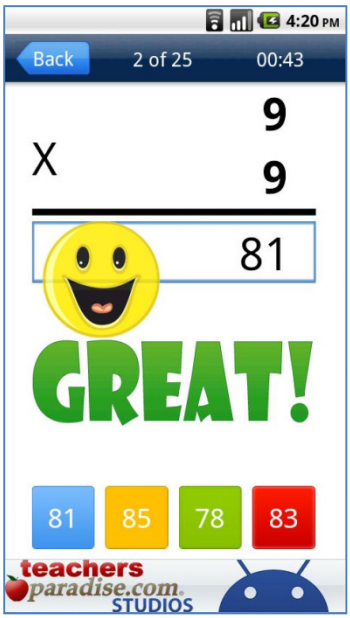


Ilustración 11: Pregunta de test de Math Practice

Su principal novedad es la posibilidad de crear nuevos cuestionarios, que permite elegir las operaciones que incluirá, el número de preguntas, habilitar sonidos, respuesta múltiple y repetir hasta acertar.

6 ANDROID

Dado que se trata de la plataforma elegida para realizar el proyecto, es conveniente introducir al lector en esta plataforma con detalle, propósito del presente capítulo.

6.1 Introducción a Android

Android es un sistema operativo nacido de la alianza de 30 organizaciones del sector de los dispositivos móviles, como fabricantes de hardware, operadores y empresas de software, comprometidos a ofrecer un mejor teléfono móvil al mercado. El resultado es un sistema operativo y entorno de desarrollo de aplicaciones capaz de ejecutarse en distintos dispositivos, lo que constituye un entorno coherente y completo para los programadores.

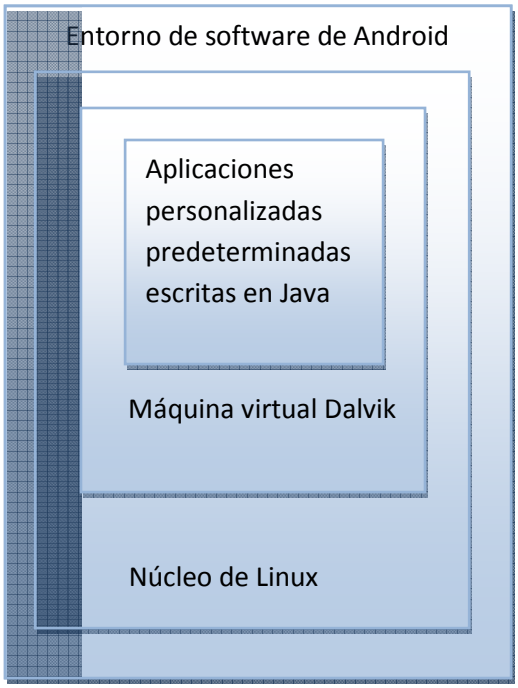
La creación de aplicaciones para Android es, en cierta medida, similar al desarrollo basado en contenedores. En lugar de una visión en la que la aplicación se ejecuta y en un punto concreto termina, Android permite que su aplicación se integre en el entorno general. Este entorno se basa en herramientas y conocimientos de Java, lo que reduce la curva de aprendizaje y proporciona la facilidad y la seguridad del desarrollo en un lenguaje gestionado. Android le permite ejecutar servicios de fondo e incluye componentes y servicios de datos que puede compartir con otras aplicaciones.

En definitiva, se trata de un entorno adecuado más que adecuado para programadores de aplicaciones. Pasaremos a continuación a hacer un repaso de la historia de Android, además de un repaso de sus principales componentes.

6.1.1 La plataforma Android

Android es un entorno de software creado para dispositivos móviles. No es una plataforma de HW. Incluye un sistema operativo basado en Linux, una completa interfaz de usuario, aplicaciones, bibliotecas de código, estructuras para aplicaciones, compatibilidad multimedia y más. Aunque los componentes del SO subyacente se escriban en C o C++, las aplicaciones para Android se diseñan en Java.

Una característica de la plataforma es que no existen diferencias entre las aplicaciones incorporadas y las creadas con el SDK. Esto significa que se pueden crear completas aplicaciones para aprovechar los recursos disponibles en el dispositivo. La siguiente figura muestra la relación entre Android y el HW sobre el que se ejecuta.



Puede que la característica más notable de Android sea su naturaleza de código abierto: la comunidad desarrolladores proporciona los elementos de los que carece. El SO basado en Linux no incorpora un entorno sofisticado pero como la plataforma es abierta, se puede modificar. Del mismo modo, se pueden obtener códecs multimedia de terceros y no es necesario depender de Google para disfrutar de nuevas funciones.

6.2 Historia de Android y evolución

La historia del sistema operativo Android nos lleva a Junio del 2005 momento en el cual Google compró una pequeña compañía cuya finalidad era el desarrollo de aplicaciones para dispositivos móviles, Android Inc. Y de misma forma Andy Rubin, uno de los cofundadores de aquella compañía, sería luego el director de la división de plataformas móviles de Google.

Ya desde ese momento empezó la especulación de que Google estaría interesada en entrar en el mercado de la telefonía móvil.

Pasaron algunos meses y empezaron a aparecer las primeras demos no oficiales y fotos clandestinas de prototipos. Se trabajó con bastante discreción hasta que el 5 de Noviembre del 2007 se anuncia la creación de la Open Handset Alliance, una organización cuyo objetivo es la difusión de la plataforma móvil Android. Fabricantes de equipos y prestadores de servicios de tecnología que unieron fuerzas para lanzar el primer sistema operativo abierto para móviles, que no estaría atado a una marca o equipo, sino que gracias a su kernel de Linux, podría ser adaptado a casi cualquier dispositivo. Para sorpresa de muchos, cinco días después del anuncio, Google lanza un Software Development Kit o SDK , que incluía un emulador de Android para ir probando las primeras líneas de código.

6.2.1 Android 1.1 a 1.5 (cupcake)

En febrero de 2009, debutó **Android 1.1,** equipado en el primer terminal Android del mundo, el **T-Mobile G1**, en España conocido como HTC Dream. A esta versión le siguió **Android 1.5 Cupcake,** que lo hizo en abril. A partir de entonces se empezarían a suceder una larga lista de versiones todas con nombres de dulces. **Cupcake** incorpora grabaciones, vídeos de Youtube, widgets y muchas otras funciones a Android.

6.2.2 Android 1.6 (donut)

Donut llega en septiembre de 2009 y con ella una mejora del **Android Market**, la tienda de aplicaciones de Android. Con Donut, Android ganó una mejor calidad de imagen y voz, búsquedas, y apoyo a las resoluciones **CDMA/EVDO, 802.1x, VPNs, WVGA.**

6.2.3 Android 2.0 – 2.1 (Eclair)

Eclair salió a finales de **octubre de 2009,** para hacer mejorar el apartado visual del S.O., y Android se hace cada vez más corporativo con la incorporación del software para gestión de correo electrónico **Microsoft Exchange**. Eclair mejora la resolución y el tamaño de las pantallas, además de la interfaz de usuario y la GUI del navegador. Por otro lado, **Eclair** introduce los fondos de pantalla animados. El 3 de diciembre de 2009 sería liberada la versión 2.01, y el 12 de enero de 2010, la versión 2.1. Ambas bajo el mismo nombre en clave.

6.2.4 Android 2.2 (Froyo)

La siguiente versión de Android fue liberada el 20 de mayo de 2010, en esta ocasión, bajo el nombre de Froyo. Destaca una optimización general del S.O, rendimiento y memoria. Entre otras novedades, incluye soporte para Flash 10.1 y Adobe Air. Desde esta versión se permite la instalación de las aplicaciones en la tarjeta SD - hasta la llegada de Froyo, éstas se instalaban en la memoria del teléfono. Entre otras características, se permite el uso del teléfono como router inalámbrico.

6.2.5 Android 2.3 (Gingerbread)

La última versión ideada para teléfonos móviles de Android (hasta la fecha) apareció el 6 de diciembre de 2010. Entre sus mejoras se incluye el soporte multi-cámara nativo, que reconoce de forma automática más de una cámara en el terminal para facilitar su configuración y resolución resultante para diversos usos tales como las video-llamadas.

Hablando de llamadas, el establecimiento de llamadas VoIP deja de ser un problema para los desarrolladores. Ahora las aplicaciones podrán ofrecer llamadas VoIP simplemente usando unas API, de manera nativa, sin tener que gestionar cuentas o demás niveles de comunicación internos.

Cambios en la interfaz gráfica y mayores facilidades para el desarrollo de juegos fueron otras de los cambios introducidos en esta última versión del S.O. para teléfonos móviles.

6.2.6 Android 3.0 (Honeycomb)

Resta una última versión por analizar. Se trata de Android 3.0 (Honeycomb) diseñada para tabletas. Se trata de una revisión del S.O. muy orientado a la navegación web y versiones de algunas de las aplicaciones más populares de Google (Gmail, Youtube) creadas específicamente para este tipo de dispositivos. Entre otras novedades, se incluye una mejora del sistema multitarea y soporte para videochat mediante GoogleTalk.

6.2.7 Historial de actualizaciones

La siguiente tabla resume las modificaciones incluidas en cada nueva versión del S.O.:

1.0	Liberado el 23 de septiembre de 2008.
1.1	Liberado el 9 de febrero de 2009.
1.5 (Cupcake)	Basado en el kernel de Linux 2.6.27. El 30 de abril de 2009, la actualización 1.5 (Cupcake) para Android fue liberada. Hubo varias características nuevas y actualizaciones en la interfaz de usuario en la actualización 1.5: • Posibilidad de grabar y reproducir videos a través del modo camcorder • Capacidad de subir videos a YouTube e imágenes a Picasa directamente desde el teléfono • Un nuevo teclado con predicción de texto • Soporte para Bluetooth A2DP y AVRCP • Capacidad de conexión automática para conectar a auricular Bluetooth a cierta distancia • Nuevos widgets y carpetas que se pueden colocar en las pantallas de inicio • Transiciones de pantalla animadas
1.6 (Donut)	Basado en el kernel de Linux 2.6.29. El 15 de septiembre de 2009, el SDK 1.6 (Donut) fue liberado. Se incluyó en esta actualización: • Una experiencia mejorada en el Android Market • Una interfaz integrada de cámara, filmadora y galería • La galería ahora permite a los usuarios seleccionar varias fotos para eliminarlas • Búsqueda por voz actualizada, con

	respuesta más rápida y mayor integración con aplicaciones nativas, incluyendo la posibilidad de marcar a contactos • Experiencia de búsqueda mejorada que permite buscar marcadores, historiales, contactos y páginas web desde la pantalla de inicio. • Actualización de soporte para CDMA/EVDO, 802.1x, VPN y text-to-speech • Soporte para resoluciones de pantalla WVGA • Mejoras de velocidad en las aplicaciones de búsqueda y cámara • Framework de gestos y herramienta de desarrollo GestureBuilder • Navegación gratuita turn-by-turn de Google
2.0 / 2.1 (Eclair)	Basado en el kernel de Linux 2.6.29. El 26 de octubre de 2009, el SDK 2.0 (Eclair) fue liberado. Los cambios incluyeron: • Velocidad de hardware optimizada • Soporte para más tamaños de pantalla y resoluciones • Interfaz de usuario renovada • Nuevo interfaz de usuario en el navegador y soporte para HTML5 • Nuevas listas de contactos • Una mejor relación de contraste para los fondos • Mejoras en Google Maps 3.1.2 • Soporte para Microsoft Exchange • Soporte integrado de flash para la cámara • Zoom digital • MotionEvent mejorado para captura de eventos multi-touch45 • Teclado virtual mejorado • Bluetooth 2.1 • Fondos de pantalla animados El SDK 2.0.1 fue liberado el 3 de diciembre de 2009. El SDK 2.1' fue liberado el 12 de enero de

	2010.
2.2 (Froyo)	Basado en el kernel de Linux 2.6.32 El 20 de mayo de 2010, el SDK 2.2 (Froyo) fue liberado. Los cambios incluyeron: • Optimización general del sistema Android, la memoria y el rendimiento • Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT • Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser • Soporte mejorado de Microsoft Exchange (reglas de seguridad, reconocimiento automático, GAL look-up, sincronización de calendario, limpieza remota) • Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y Browser • Funcionalidad de Wi-Fi hotspot y thetering por USB • Permite desactivar el tráfico de datos a través de la red del operador • Actualización del Market con actualizaciones automáticas • Cambio rápido entre múltiples idiomas de teclado y sus diccionarios • Marcación por voz y compartir contactos por Bluetooth • Soporte para contraseñas numéricas y alfanuméricas • Soporte para campos de carga de archivos en la aplicación Browser • Soporte para la instalación de aplicación en la memoria expandible • Soporte para Adobe Flash 10.1 • Soporte para pantallas de alto número de Puntos por pulgada, tales como 4" 720p
2.3 (Gingerbread)	Basado en el kernel de Linux 2.6.35.7 Actual en smat

	El 6 de diciembre de 2010, el SDK 2.3 (Gingerbread) fue liberado. Los cambios incluyeron: • Actualización del diseño de la interfaz de usuario • Soporte para pantallas extra grandes y resoluciones WXGA y mayores • Soporte nativo para telefonía VoIP SIP • Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC • Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves • Soporte para Near Field Communication • Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema • Teclado multi-táctil rediseñado • Soporte mejorado para desarrollo de código nativo • Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos • Recolección de elementos concurrentes para un mayor rendimiento • Soporte nativo para más sensores (como giroscopios y barómetros) • Un administrador de descargas para descargar archivos grandes • Administración de la energía mejorada y control de aplicaciones mediante la administrador de tareas • Soporte nativo para múltiples cámaras • Cambio de sistema de archivos de YAFFS a ext4
3.0 (Honeycomb)	• Mejor soporte para tabletas • Escritorio 3D con widgets rediseñados • Sistema multitarea mejorado • Mejoras en el navegador web

	predeterminado, entre lo que destaca la navegación por pestañas, auto-relleno de formularios, sincronización de favoritos con Google Chrome y navegación privada • Soporte para videochat mediante Google Talk

Tabla 1: Versiones de Android

6.2.8 Fragmentación

Las diferentes versiones de Android coexisten actualmente en el mercado, dando lugar a lo que se ha dado en llamar fragmentación. Esto es, es el fenómeno por el que la división entre versiones condiciona la experiencia de uso final, ya que en función de la versión de Android que porte nuestro móvil nos encontraremos con que nuestro Market incluye o no ciertas cosas, y que algunas funciones se encuentran restringidas o simplemente no existen (como la aparición del tethering con la versión 2.2).

Éste fenómeno se fue acentuando con el paso de las versiones, quedando los dispositivos con las primeras versiones relegados a meras anécdotas en la historia de un sistema operativo cada vez más completo, pues su funcionalidad se vio seriamente truncada. El gráfico a continuación muestra la distribución de las diferentes versiones en el mercado:

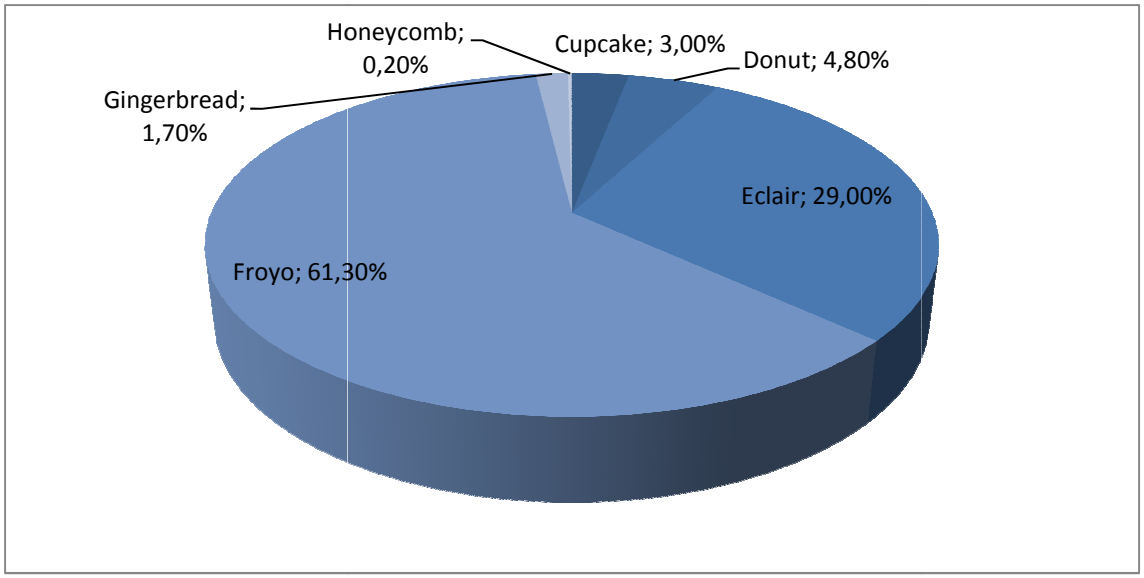


Ilustración 12: Versiones de Android en el mercado

6.3 Arquitectura

En la siguiente figura podemos ver un diagrama que muestra los principales componentes de la arquitectura de Android:

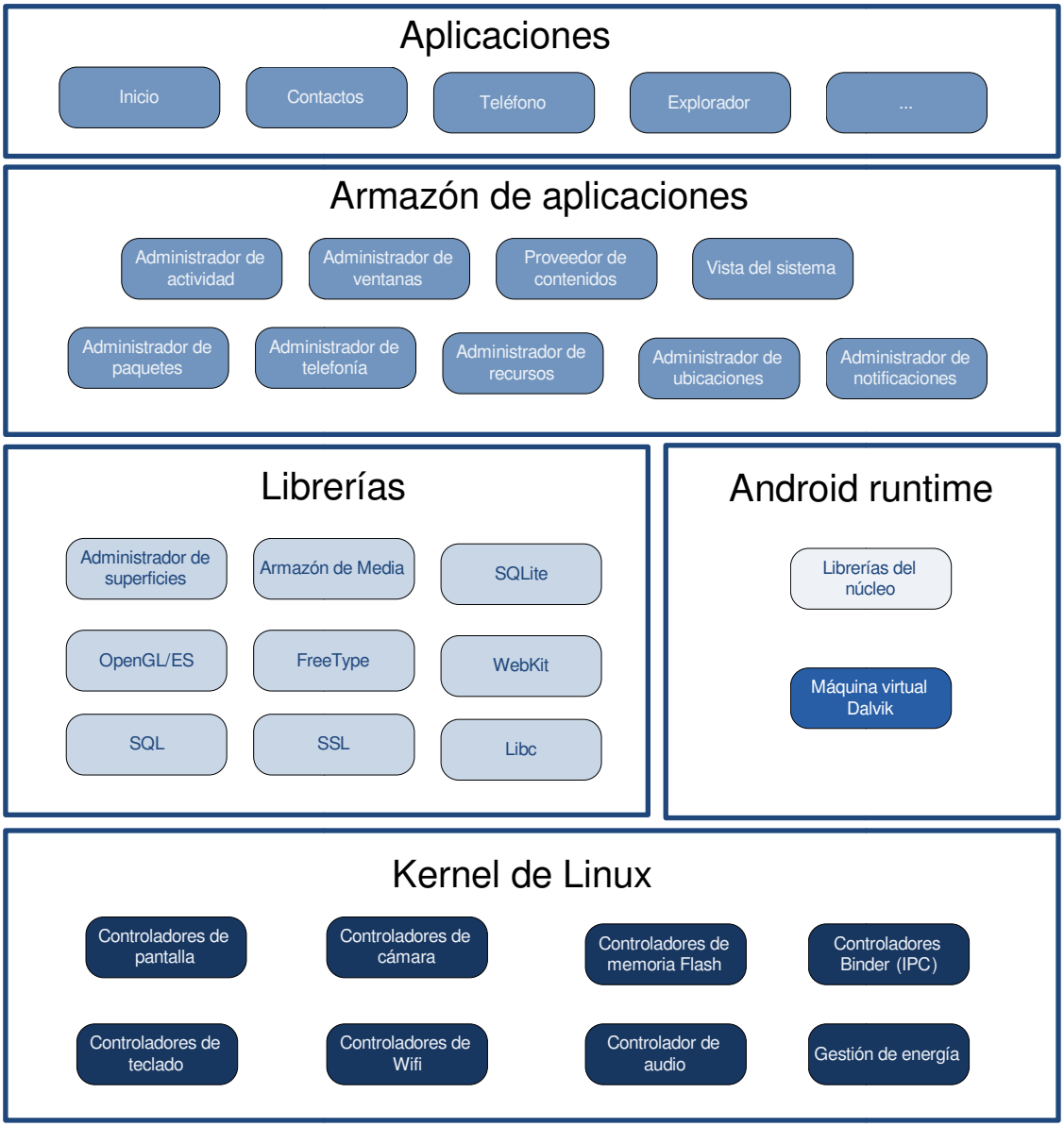


Ilustración 13: Arquitectura de Android

6.3.1 Dalvik

Android utiliza una máquina virtual propia llamada Dalvik, que ha sido diseñada para optimizar la memoria y los recursos de hardware en el entorno de los teléfonos móviles. A diferencia de la máquina virtual de Java, basada en el uso de pilas, la máquina virtual Dalvik está basada en registros.

Dalvik es una máquina virtual intérprete que ejecuta archivos en el formato Dalvik Executable (*.dex), un formato optimizado para el almacenamiento eficiente y ejecución mapeable en memoria. Su objetivo fundamental es el mismo que cualquier máquina virtual, permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar, y la máquina virtual interpreta este bytecode a la hora de ejecutar el programa.

El hecho de no utilizar la máquina virtual de Java es la necesidad de optimizar al máximo los recursos y enfocar el funcionamiento de los programas hacia un entorno de escasos recursos de memoria, procesador y almacenamiento como es el que se tiene en los teléfonos.

Dalvik está basada en registros y puede ejecutar clases compiladas (válgase la redundancia) por un compilador Java y que posteriormente han sido convertidas al formato nativo usando la herramienta “dx”. Hay que decir que esta máquina virtual corre por encima de un kernel Linux 2.6, el cuál le permite, entre otras cosas, delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel. La Dalvik VM ha sido optimizada también para que haya múltiples instancias suyas funcionando con un impacto muy bajo en el rendimiento de la memoria del dispositivo. Este aspecto de usar varias máquinas virtuales se pensó para proteger a las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma a las demás.

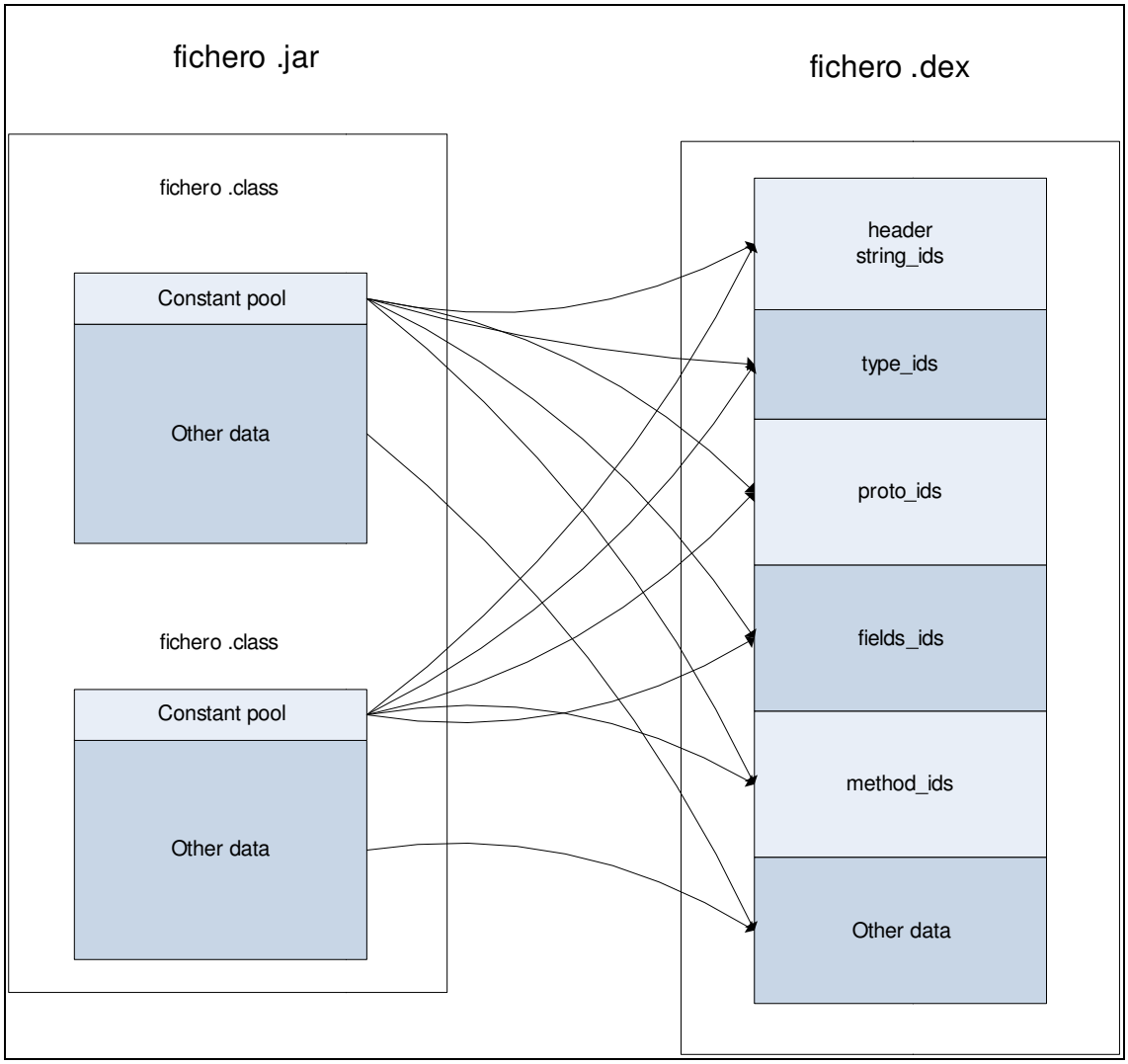


Ilustración 14: Ejemplo de fichero .dex

La figura anterior nos muestra como en los ficheros ejecutables de Dalvik se agrupan los datos de las clases que componen la aplicación con el objetivo anteriormente mencionado de ocupar una cantidad menor de memoria, ya que no se almacenan los datos que puedan estar duplicados en las distintas clases. Por otra parte una vez instalados en un dispositivo, los ficheros ejecutables pueden sufrir otras modificaciones como por ejemplo que se altere el orden de los bytes de ciertos datos con el fin de optimizar aún más el rendimiento del terminal cuando ejecute la aplicación.

6.3.2 Componentes

En este apartado hablaremos sobre los componentes o bloques que podemos encontrar en cualquier aplicación de Android. No es obligatorio implementarlos todos, por lo que podríamos decir que las aplicaciones están compuestas por una combinación libre de los bloques o componentes descritos a continuación. Todos estos componentes deben ser declarados de forma explícita en el fichero AndroidManifest.xml donde se encuentran definidos otros datos importantes como los permisos, siendo un fichero básico en cualquier aplicación.

6.3.2.1 Activity

En Android, una aplicación puede disponer o no de interfaz de usuario. En caso de hacerlo, ha de albergar al menos un elemento Activity.

Se trata del componente principal de una GUI en Android. Y asociarlo a cada pantalla de una aplicación servirá para tener una de su función.

Una Activity refleja una determinada actividad llevada a cabo por una aplicación, lleva asociada la interfaz de usuario representada por la clase View y sus derivados y responde a eventos iniciados por el sistema o el usuario.

En la mayoría de los casos, las aplicaciones se componen de más de una pantalla, por lo que incluirán más de una actividad.

6.3.2.2 Intent

El paso de una pantalla a otra se consigue mediante la inicialización de una nueva actividad, y en este punto entra el concepto de Intent. Se puede identificar Intent cómo la intención de realizar una acción (el comienzo de una nueva actividad). Al iniciar un intent, una aplicación puede lanzar otra aplicación capaz de satisfacer la intención de la actividad que lo inició (ejemplo escribir un mensaje de texto, enviar un sms, mostrar una imagen, compartir un fichero...). Es posible definir explícitamente que actividad ha de atender el Intent. De esta forma, nos ayudará a desplazarnos entre actividades (pantallas) dentro de una aplicación.

6.3.2.3 Service

Al definir Activity hablamos de aplicaciones de Android sin GUI. Pues bien, el estas aplicaciones, son los servicios. En concepto, son exactamente iguales a los servicios o demonios presentes en cualquier otro sistema operativo, ejecutándose en segundo plano, sin que el usuario tenga que ser consciente de ello. Entre las acciones de las actividades encontramos, por ejemplo: la actualización de datos, mostrar notificaciones...

6.3.2.4 Broadcast receiver

Un Broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema.

Existe la posibilidad de generar mensajes Broadcast en tu propia aplicación que estén dirigidos a cualquier Broadcast receiver que se encuentre activo mediante Intents. Al igual que los servicios, el componente Broadcast receiver carece de interfaz gráfica. Pero pueden mostrarse en la pantalla haciendo uso de una API para notificaciones.

6.3.2.5 Content provider

Este componente es el encargado de compartir información entre aplicaciones (ocultando detalles como su almacenamiento interno, estructura, o implementación.

En Android existe una colección de contents provider predefinidos y ya implementados que permiten compartir información tal como contactos, imágenes, sms...

Los content provider son objetos de la clase Content Provider, que se encuentra localizada en el paquete android.content. Cada objeto de Content provider se encuentra asociado a una URI única que le identifica y le hace accesible para el resto de aplicaciones. Los datos que se compartan mediante content provider se almacenan a menudo como una tabla de la base de datos SQLite, que describiré en secciones posteriores.

6.3.3 Ciclo de vida de una aplicación

Cada aplicación Android es ejecutada en su propio proceso. Este proceso es creado cuando se hace necesaria la ejecución de su código y continuará vivo hasta el sistema reclame su memoria para asignársela a otra aplicación. Es ésta una característica distintiva de Android que merece ser comentada. Es el Sistema operativo el encargado de controlar el ciclo de vida de un proceso y no la aplicación. Para ello, Android se basa en su importancia (la aplicación más importante es la que se está mostrando en primer plano en el dispositivo), los recursos disponibles... todo ello es transparente al usuario.

A continuación se muestra el ciclo de vida del principal componente Android: Activity.

6.3.3.1 Ciclo de vida del componente Activity

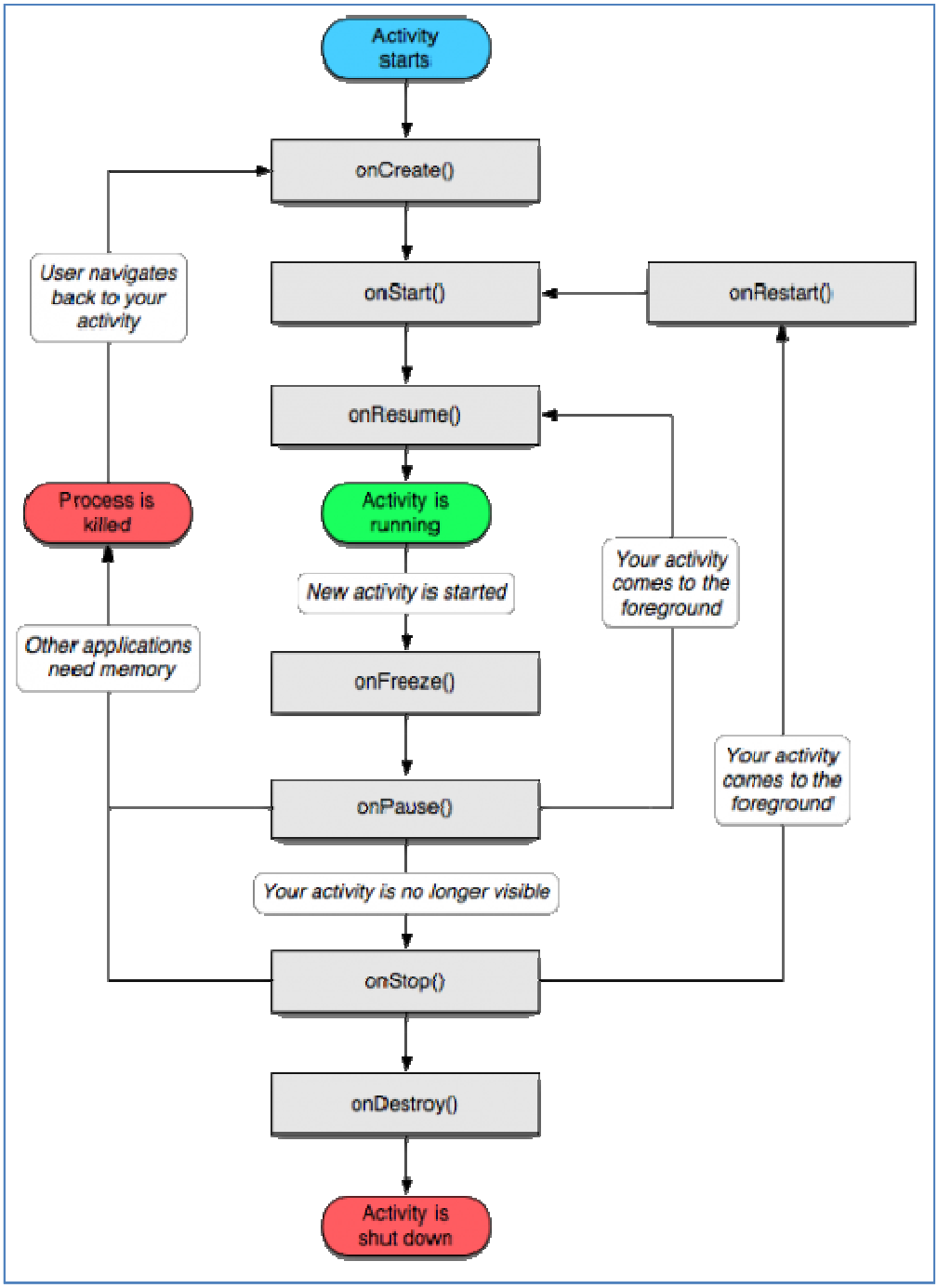
Android maneja las actividades como una pila, la creación de una nueva actividad la sitúa en la cima de la pila, quedando delante de la actividad anterior. Como un usuario puede cambiar de actividad en cualquier momento, afectando a los recursos generales del sistema, todas las clases Activity han de admitir ser detenidas y terminadas en un momento dado. Para administrar este escenario, las aplicaciones Android y las Activity han de diseñarse de la siguiente forma: Por métodos relacionados con eventos – definidos en Activity- se puede configurar y finalizar su estado. Los principales métodos del ciclo vital de Activity se definen a continuación:

Estado	Descripción	Abortable?	Siguiente
onCreate()	Se ejecuta cuando se crea la Activity por primera vez. Aquí es donde se deberían crear views, enlazar datos a listas, en definitiva el proceso de inicialización de nuestra aplicación.	No	onStart()
onRestart()	Se ejecuta cuando la aplicación se ha cerrado y se va a ejecutar nuevamente.	No	onStart()
onStart()	Se ejecuta cuando al aplicación aparece visible para el usuario. Si la aplicación es un proceso en background el siguiente estado es onStop(), si la aplicación se ejecuta en foreground el siguiente método es onResume().	No	onResume()o onStop()
onResume()	Se ejecuta cuando la Activity interactúa con el usuario. En éste punto la Activity está en la cima de la pila.	No	onPause()
onPause()	Se ejecuta cuando el sistema está a punto de continuar una Activity anterior. Se utiliza típicamente para guardar datos que no se han grabado anteriormente, parar animaciones y otras acciones que consuman CPU. Seguida por onResume() si la actividad vuelve a primer plano o onStop() si es invisible para el usuario.	**Si**	onResume()o onStop()

onStop()	Se ejecuta cuando la Activity deja de ser visible al usuario, porque otra Activity ha continuado y pasa a un lugar más prioritario de la pila. Puede ocurrir porque una nueva Activity ha sido creada, una Activity ya creada pasa a primer plano o ésta está siendo destruida. El siguiente método será onRestart() si la Activity vuelve a interaccionar con el usuario o por onDestroy() si la Activity será destruida.	**Si**	onRestart()o onDestroy()
onDestroy()	Última llamada antes de destruir la Activity. Puede ocurrir porque la actividad está acabando (llamada a finish()) o porque el sistema destruirá la instancia para guardar espacio. Se puede distinguir esos escenarios con el método isFinishing().	**Si**	*nada*

Tabla 2: Métodos del ciclo de vida de una Actividad

El siguiente diagrama muestra la relación y transiciones entre estados para una Activity:



6.4 Aspectos del desarrollo de una aplicación

En el apartado anterior, Arquitectura Android, hemos comprendido los elementos que vertebran el sistema operativo, cómo las actividades suponen el elemento básico que constituye las aplicaciones. En este apartado se van a tratar diversos aspectos del sistema operativo: La base de datos de serie, la conexión de red en las aplicaciones y la organización de un proyecto. El conocimiento de estas características será en más que necesario para el desarrollador que se embarque en la creación de aplicaciones para Android.

6.4.1 Base de datos

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite.

SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y por supuesto ser de código libre.

SQLite no tiene todas las funciones de los productos de base de datos cliente/servidor comerciales, pero ofrece todo lo necesario para almacenamiento local de datos, a la vez que resulta rápido y sencillo.

6.4.1.1 Sqlite3

Al crear una base de datos para una aplicación en Android, los archivos de la misma se crean en /data/data/[NOMBRE_PAQUETE]/databases/db.nombre. Estos archivos son SQLite, pero existe una forma de manipular, volcar, restaurar y trabajar con bases de datos a través de los mismos mediante la herramienta incluida en el SDK de Android, sqlite3.

Se trata de una herramienta de línea de comandos desde la que se pueden ejecutar comandos básicos como SELECT o INSERT, u otros más avanzados para tablas (CREATE o ALTER). También resulta útil para volcar y abrir datos (mediante los comandos .dump y .load, respectivamente).

6.4.2 Conexión de Red

En los teléfonos Android existen, básicamente, 2 tipos de redes de datos: las 2G y las 3G.

Las redes 2G son las más usadas; normalmente usadas para enviar y recibir llamadas, aunque también pueden usarse para transmitir datos. Así, dentro de las 2G encontramos el GPRS y el EDGE (E-GPRS). En cambio, las 3G son usadas (además de para transmitir voz) para la transmisión de datos a gran velocidad, encontrando dentro de éstas las redes UMTS y las HSPA.

A continuación tenemos los iconos de conexión que puede presentar el teléfono Android, y que se describe de derecha a izquierda:

Ilustración 16: Iconos de conexión en un teléfono Android

- Conexión 2G sin datos: Con esta conexión no se podrá usar los servicios de transmisión de datos como Internet, correo electrónico...
- Conexión 2G GPRS: es la conexión de datos más lenta, de unos 6 KB/s (equivalente a una conexión módem antigua de 56 Kbps).
- Conexión 2.5G (EDGE): es una mejora del GPRS, que permite navegar a velocidades más aceptables que el GPRS, pero sin llegar, ni de lejos, a las conseguidas por las 3G.
- Conexión 3G (UMTS): 3G y UMTS son palabas sinónimas; tiene una cobertura de entre el 70 y el 85% y te permite navegar a alta velocidad.
- Conexión 3.5G (HSPA): la tecnología HSPA es la más rápida de la actualidad (en el mercado español) y permite velocidades de hasta 7Mbps de bajada (HSDPA) y 2 Mbps de subida (HSUPA). Está presente en los núcleos urbanos sobretodo y se representa en el teléfono mediante el icono H

6.4.3 Organización de Proyecto

A continuación, vamos a hacer un resumen de la estructura general de un proyecto en Android.

Cuando creamos un nuevo proyecto Android en Eclipse se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación. Esta estructura será común a cualquier aplicación, independientemente de su tamaño y complejidad.

En la siguiente imagen vemos los elementos creados inicialmente para un nuevo proyecto Android:

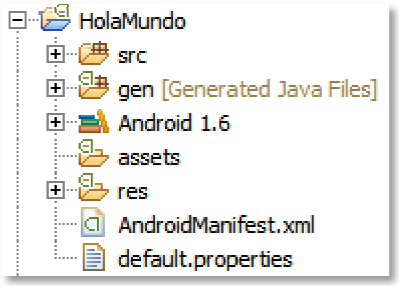


Ilustración 17: Carpetas en un proyecto Android

A continuación describimos los elementos principales.

6.4.3.1 Carpeta /src/

Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity) principal de la aplicación, siempre bajo la estructura del paquete java definido.

Ilustración 18: Ejemplo carpeta src

6.4.3.2 Carpeta /res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:

- /res/drawable/. Contienen las imágenes de la aplicación. Se puede dividir en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi para utilizar diferentes recursos dependiendo de la resolución del dispositivo.
- /res/layout/ Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land para definir distintos layouts dependiendo de la orientación del dispositivo.
- /res/anim/ Contiene la definición de las animaciones utilizadas por la aplicación.
- /res/menu/ Contiene la definición de los menús de la aplicación.
- /res/values/ Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), etc.
- /res/xml/ Contiene los ficheros XML utilizados por la aplicación.
- /res/raw/ Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

Como ejemplo, para un proyecto nuevo Android, se crean los siguientes recursos para la aplicación:

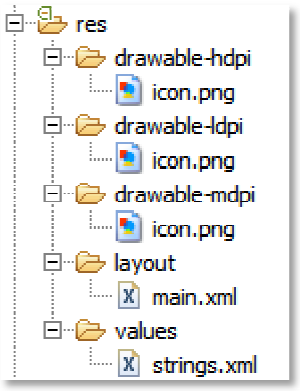


Ilustración 19: Ejemplo carpeta res

6.4.3.3 Carpeta /gen/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en Java, dirigidos al control de los recursos de la aplicación.

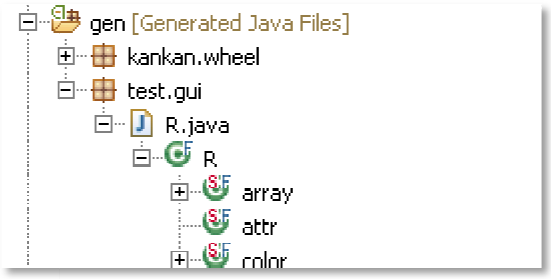


Ilustración 20: carpeta gen

El más importante es el que se puede observar en la imagen, el fichero R.java, y la clase R. Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código a través de este dato. Así, por ejemplo, la constante R.drawable.icon contendrá el ID de la imagen "icon.png" contenida en la carpeta /res/drawable/. Veamos como ejemplo la clase R creada por defecto para un proyecto nuevo:

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Ilustración 21: Clase R.java

6.4.3.4 Carpeta /assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, de datos, etc.

La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Usaremos uno u otro según las necesidades de nuestra aplicación.

6.4.3.5 Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, …), sus componentes (pantallas, mensajes, …), o los permisos necesarios para su ejecución. Veremos más adelante más detalles de este fichero.

6.4.4 GUI

Al hablar de Activity ya mencionamos que, además de desempeñar un papel fundamental en su ciclo de vida, forman las pantallas de la aplicación. A este apartado corresponde explicar cómo se organizan y crean estas pantallas. Cada actividad está formada por componentes secundarios llamados vistas.

Las vistas son los componentes que ven los usuarios, y con los que interactúan. Se encargan del diseño de la vista, proporcionan elementos de entrada al usuario, botones, formularios… Una colección jerárquica de vistas se emplea para formar la actividad. Las vistas emplean cadenas de texto, colores, estilos e imágenes que son compilados y ofrecidos como recursos para las aplicaciones – a estos recursos se accede mediante la clase R.java, generada automáticamente.

El siguiente diagrama muestra una relación entre Actividades, vistas y recursos:

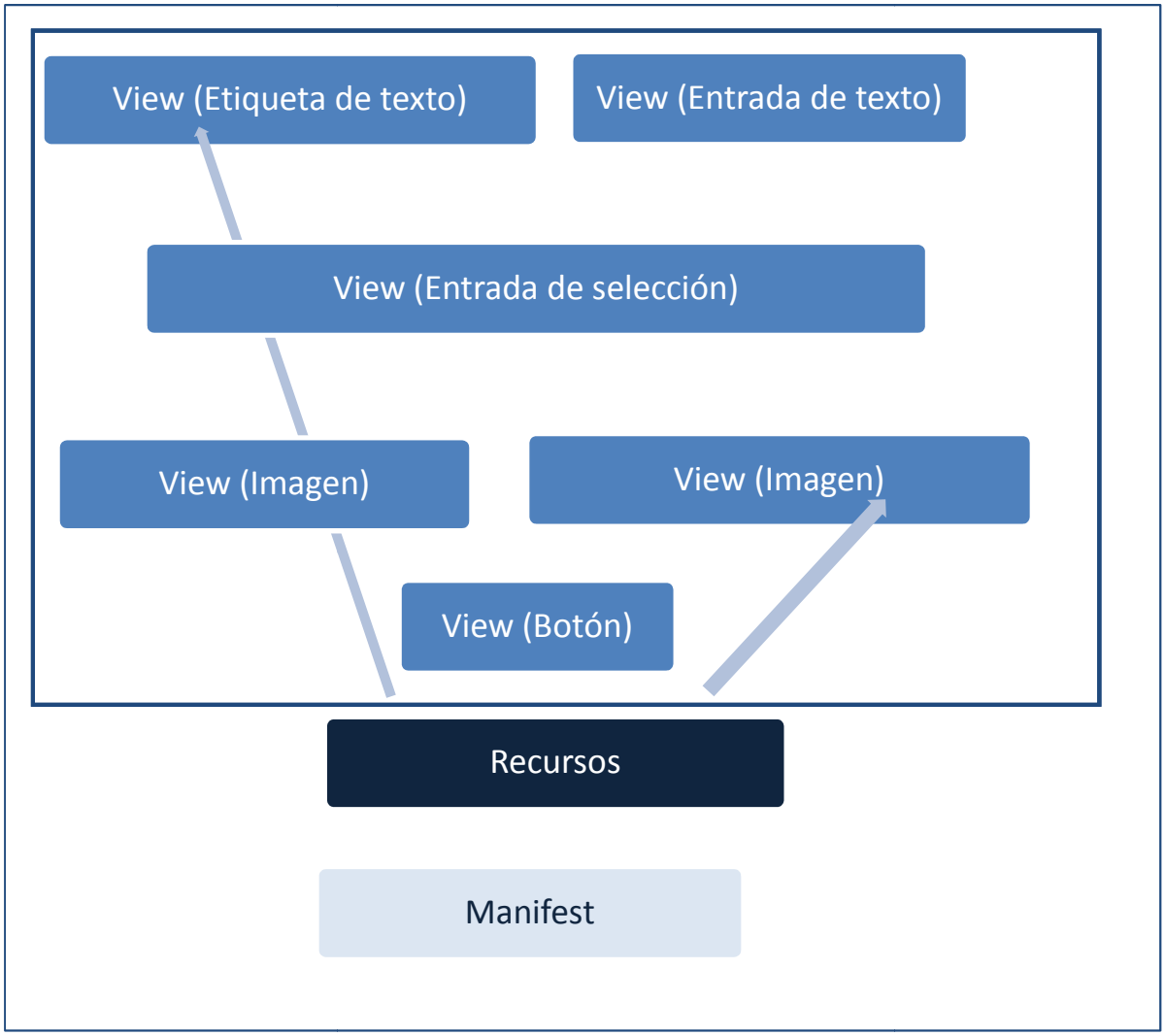


Ilustración 22: Relación entre activity, view, Manifest y recursos

Los elementos de las vistas se incluyen en una actividad con un diseño dado formando una pantalla.

Las vistas y los diseños se pueden definir directamente en código (programáticamente):

```
LinearLayout linear;

TextView text;

linear = new LinearLayout(this);

linear.setOrientation(LinearLayout.VERTICAL);

text = new TextView(this);


text.setText("This is an example for the Bright Hub!");

linear.addView(text);

setContentView(linear);
```

Ilustración 23: definición programática de vistas

O en un archivo de recurso XML:

```
<?xml version="1.0" encoding="windows-1250"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout_width="fill_parent"

android:layout_height="fill_parent">

<TextView

android:id="@+android:id/textView"

android:layout_width="fill_parent"

android:layout_height="wrap_content"/>

</LinearLayout>
```

Ilustración 24: Definición de vistas mediante XML

El resultado obtenido, para los ejemplos mostrados, es el mismo.

Es habitual hacer la asignación de las vistas a la Actividad dentro del método onCreate(), asociando un archivo de diseño XML al método setContentView(). Aunque como acabamos de ver no es imprescindible, suele ser más sencillo utilizar un recurso de diseño XML en cada actividad. Las vistas definidas en el fichero XML se disponen formando un árbol.

6.4.4.1 Layouts

En el apartado anterior hemos hecho mención a Vistas encargadas del diseño de la pantalla. Bien, estas vistas son los Layouts, elementos no visuales destinados a controlar la distribución, posición y dimensiones de los controles que se insertan en su interior (cuadros de texto, botones, etiquetas de texto, imágenes, incluso otros layouts). Estos componentes extienden a la clase base ViewGroup, como muchos otros componentes contenedores, es decir, capaces de contener a otros controles. Android nos proporciona una colección de Layouts con propiedades diversas, que se exponen a continuación.

6.4.4.1.1 FrameLayout

Éste es el más simple de todos los layouts de Android. Un FrameLayout coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente (a menos que éste último tenga transparencia). Por ello, suele utilizarse para mostrar un único control en su interior, a modo de contenedor sencillo para un sólo elemento sustituible, por ejemplo una imagen.

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <EditText android:id="@+id/TxtNombre"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</FrameLayout>
```

Ilustración 25: Ejemplo de FrameLayout

6.4.4.1.2 LinearLayout

El siguiente layout Android en cuanto a nivel de complejidad es el LinearLayout. Este layout apila uno tras otro todos sus elementos hijos de forma horizontal o vertical (según se establezca su propiedadandroid:orientation). Al igual que en un FrameLayout, los elementos contenidos en un LinearLayout pueden establecer sus propiedades de ancho y alto para determinar sus dimensiones dentro del layout. Pero en el caso de un LinearLayout, tendremos otro parámetro con el que jugar, la propiedad android:layout_weight. Esta propiedad nos va a permitir dar a los elementos contenidos en el layout unas dimensiones proporcionales entre ellas.

Así pues, a pesar de la simplicidad aparente de este layout resulta ser lo suficiente versátil como para sernos de utilidad en muchas ocasiones.

A continuación, un ejemplo:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <EditText android:id="@+id/TxtDato1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1" />

    <EditText android:id="@+id/TxtDato2"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="2" />

</LinearLayout>
```

Ilustración 26: Ejemplo de LinearLayout

6.4.4.1.3 TableLayout

Un TableLayout permite distribuir sus elementos hijos de forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla.
La estructura de la tabla se define de forma similar a como se hace en HTML, es decir, indicando las filas que compondrán la tabla (objetos TableRow), y dentro de cada fila las columnas necesarias, con la salvedad de que no existe ningún objeto especial para definir una columna (algo así como un *TableColumn*) sino que directamente insertaremos los controles necesarios dentro del TableRow y cada componente insertado (que puede ser un control sencillo o incluso otro ViewGroup) corresponderá a una columna de la tabla. De esta forma, el número final de filas de la tabla se corresponderá con el número de elementos TableRowinsertados, y el número total de columnas quedará determinado por el número de componentes de la fila que más componentes contenga.

Por norma general, el ancho de cada columna se corresponderá con el ancho del mayor componente de dicha columna, pero existen una serie de propiedades que nos ayudarán a modificar este comportamiento:

android:stretchColumns. Indicará las columnas que pueden expandir para absorber el espacio libre dejado por las demás columnas a la derecha de la pantalla.

android:shrinkColumns. Indicará las columnas que se pueden contraer para dejar espacio al resto de columnas que se puedan salir por la derecha de la pantalla.

android:collapseColumns. Indicará las columnas de la tabla que se quieren ocultar completamente.

Todas estas propiedades del TableLayout pueden recibir una lista de índices de columnas separados por comas (ejemplo: android:stretchColumns=”1,2,3″) o un asterisco para indicar que debe aplicar a todas las columnas (ejemplo: android:stretchColumns=”*”).

Otra característica importante es la posibilidad de que una celda determinada pueda ocupar el espacio de varias columnas de la tabla (análogo al atributo colspan de HTML). Esto se indicará mediante la propiedad android:layout_span del componente concreto que deberá tomar dicho espacio.

Veamos un ejemplo con varios de estos elementos:

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1" >

    <TableRow>
        <TextView android:text="Celda 1.1" />
        <TextView android:text="Celda 1.2" />
    </TableRow>

    <TableRow>
        <TextView android:text="Celda 2.1" />
        <TextView android:text="Celda 2.2" />
    </TableRow>

    <TableRow>
        <TextView android:text="Celda 3"
                  android:layout_span="2" />
    </TableRow>

</TableLayout>
```

6.4.4.1.4 RelativeLayout

El último tipo de layout que vamos a ver es el RelativeLayout. Su principal característica es la de permitir especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio layout. Así, al incluir un nuevo elemento X podremos indicar por ejemplo que debe ***colocarse*** **debajo del elemento Y *y* alineado a la derecha del layout padre.** Veamos esto en el ejemplo siguiente:

```
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <EditText android:id="@+id/TxtNombre"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content" />

        <Button android:id="@+id/BtnAceptar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/TxtNombre"
            android:layout_alignParentRight="true" />

</RelativeLayout>
```

Ilustración 27: Ejemplo de RelativeLayout

Al igual que estas tres propiedades, en un RelativeLayout tendremos un sinfín de propiedades para colocar cada control justo donde queramos: Podemos definir la posición de un elemento relativa a otro control, al layout padre. O establecer opciones de margen y relleno.

6.4.4.2 Controles básicos

Después del repaso por los principales layouts que Android nos ofrece para distribuir los controles de la interfaz en la pantalla, vamos a enumerar y describir de forma breve los principales controles disponibles.

6.4.4.2.1 Button e ImageButton

Un control de tipo Button es el botón más básico que podemos utilizar. Se trata del típico botón que puede mostrar un texto (button) o una imagen(imageButton) - existe la opción de combinar ambas, mediante las propiedades drawableTop, drawableBottom, drawableRight y drawableLeft. Es posible asociar 'listener' a los principales eventos del botón, (clic en el botón, pulsado o liberar pulsación).

En el ejemplo siguiente definimos un botón con el texto "Púlsame" asignando su propiedad android:text. Además de esta propiedad podríamos utilizar muchas otras como el color de fondo (android:background), estilo de fuente (android:typeface), color de fuente (android:textcolor), tamaño de fuente (android:textSize), etc.

```
<Button android:id="@+id/BtnBoton1"
    android:text="Púlsame"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Ilustración 28: Uso de button en fichero XML

6.4.4.2.2 ToggleButton

Un control de tipo ToggleButton es un tipo de botón que puede permanecer en dos estados, pulsado/no pulsado. En este caso, en vez de definir un sólo texto para el control definiremos dos, dependiendo de su estado. Así, podremos asignar las propiedades android:textOn yandroid:textoOff para definir ambos textos. Veamos un ejemplo a continuación:

```
<ToggleButton android:id="@+id/BtnBoton2"
    android:textOn="ON"
    android:textOff="OFF"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Ilustración 29: Uso de togglebutton en fichero XML

6.4.4.2.3 ImageView

El control ImageView permite mostrar imágenes en la aplicación. La propiedad más interesante es android:src, que permite indicar la imagen a mostrar. Lo normal será indicar como origen de la imagen el identificador de un recurso de nuestra carpeta /res/drawable, por ejemplo: **android:src="@drawable/unaimagen"**. Además de esta propiedad, existen algunas otras útiles en algunas ocasiones como las destinadas a establecer el tamaño máximo que puede ocupar la imagen,android:maxWidth y android:maxHeight.

```
<ImageView android:id="@+id/ImgFoto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />
```

Ilustración 30: Uso de imageview en fichero XML

6.4.4.2.4 TextView

El control TextView es otro de los clásicos en la programación de GUIs, las etiquetas de texto, y se utiliza para mostrar un determinado texto al usuario. Al igual que en el caso de los botones, el texto del control se establece mediante la propiedad android:text.

Aparte de esta propiedad, la naturaleza del control hace que las más interesantes sean las que establecen el formato del texto mostrado, que al igual que en el caso de los botones son las siguientes: android:background (color de fondo), android:textColor (color del texto), android:textSize (tamaño de la fuente) y android:typeface (estilo del texto: negrita, cursiva, …).

```
<TextView android:id="@+id/LblEtiqueta"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Escribe algo:"
    android:background="#AA44FF"
    android:typeface="monospace" />
```

Ilustración 31: Uso de textview en fichero XML

6.4.4.2.5 EditText

El control EditText es el componente de edición de texto que proporciona la plataforma Android. Permite la introducción y edición de texto por parte del usuario, por lo que en tiempo de diseño la propiedad más interesante a establecer, además de su posición/tamaño y formato, es el texto a mostrar, atributoandroid:text.

```
<EditText android:id="@+id/TxtTexto"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/LblEtiqueta" />
```

Ilustración 32: Uso de edittext en fichero XML

7 ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN 'TESTING '

En este capítulo se dedica a detallar cada una de las siguientes etapas del proceso de desarrollo: Análisis, diseño e implementación. De esta forma, se logra una visión más concreta e informada del proyecto.

7.1 Análisis

En este apartado se especificarán los requisitos de usuario además de establecerse varios casos de uso.

7.1.1 Identificación del entorno tecnológico

En este punto se analizan las necesidades tecnológicas del sistema. Para determinar la mejor solución se ha realizado un pequeño estudio sobre qué opción es la idónea tanto para el desarrollador como para el usuario. A continuación presentamos las opciones elegidas teniendo la certeza que cubrirán las necesidades básicas.

Entorno tecnológico de usuario

Dispositivo Móvil	Versión Android 2.0 o superior

Tabla 3: Entorno tecnológico del usuario

Entorno tecnológico del desarrollador

Equipo (Servidor de prueba)	Intel Core 2 CPU 6400 2 GB RAM HD 500 GB
S.O.	Windows XP Media Center SP3
IDE	Eclipse Helios + Android Plugin Netbeans 6.5

SDK Android	Ver 2.0
Dispositivo móvil	HTD Desire HD (Android 2.2), HTC Hero (Android 2.1), HTC Legend (Android 2.2)

Tabla 4: Entorno tecnológico del desarrollador

Como podemos observar, se ha decidido desarrollar la aplicación enfocando el trabajo a la versión 2.0 del sistema operativo. Aunque existen versiones superiores instaladas en dispositivos más modernos que ofrecen al programador mayores capacidades, se ha decidido programar para la versión 2.0 convirtiendo la aplicación en compatible con cualquier versión posterior. Se ha tomado esta decisión ya que elegir esta versión permitirá que la aplicación sea compatible con el 95% de los dispositivos Android.

7.1.2 Requisitos software

En este apartado se especifican los principales requisitos de software del sistema. La formulación de dichos requisitos se ha llevado a cabo en formato tabular. Los requisitos software nos proporcionarán una visión completa del comportamiento del sistema. Los campos para definir dichos requisitos de dividen en:

- **Título**: Breve descripción del objetivo del requisito. No se indicará en la tabla como campo, sino que formará parte del título de la misma.
- **Identificador**: Identificación unívoca del requisito. Debe seguir la nomenclatura RSW-XX, dónde 'X' es un número entre 0 y 0, y se incrementarán en una unidad con cada nuevo requisito.
- **Tipo:** funcionales, de rendimiento o de usabilidad.
- **Descripción**: Especifica el significado del requisito.
- Estabilidad: Hace referencia a la sensibilidad del requisito a ser modificado. Toma los valores 'estable' o 'no estable'.
- **Prioridad**: Define la prioridad con la que debe ser resuelto dicho requisito.

Puede tomar los valores de "Alta", "Media" y "Baja".

Identificación de usuarios			
identificador	RSW-01		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	El sistema permitirá a los usuarios identificarse en la aplicación a partir de un su número de identificador de alumno (NIA)		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Listar cuestionarios			
identificador	RSW-02		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	La aplicación móvil ofrecerá al usuario los cuestionarios disponibles en forma de lista, permitiendo clasificarlos por asignatura.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Descarga de cuestionarios			
identificador	RSW-03		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	La aplicación móvil descargará al inicio los nuevos cuestionarios disponibles en el servidor para el usuario identificado en la aplicación.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Realizar tests			
identificador	RSW-04		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	El sistema permitirá al usuario ejecutar los tests almacenados en el dispositivo móvil.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Modo de ejecución			
identificador	RSW-05		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	Existirán dos modos de ejecución de tests: - Entrenamiento: Las preguntas se corrigen de forma individual. - Examen: El test se corrige tras completarlo.		
Estabilidad	☐ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Límite de tiempo			
identificador	RSW-06		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	Se podrá fijar un límite de tiempo desde la aplicación para realizar los tests.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Personalizar tests			
identificador	RSW-07		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	El usuario podrá crear tests con un número de preguntas por él seleccionable para sus asignaturas.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Interfaz intuitiva			
identificador	RSW-08		
Tipo	☒ funcional	☐ rendimiento	☐ usabilidad
Descripción	Los usuarios del sistema aprenderán el manejo de la aplicación de forma rápida y sin necesidad de periodo de formación.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tiempo de espera del cliente			
identificador	RSW-09		
Tipo	☐ funcional	☒ rendimiento	☐ usabilidad
Descripción	El tiempo de espera del cliente para conectar al servidor será de 5 segundos.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tiempo de espera BD			
identificador	RSW-10		
Tipo	☐ funcional	☒ rendimiento	☐ usabilidad
Descripción	El tiempo de respuesta de la aplicación al actualizar o acceder a la base de datos deberá ser de cinco segundos como máximo.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Disponibilidad de la aplicación			
identificador	RSW-11		
Tipo	☐ funcional	☒ rendimiento	☐ usabilidad
Descripción	El servidor desarrollado servirá únicamente para probar la aplicación cliente, por lo que no se garantiza su disponibilidad.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Recuperación ante errores			
identificador	RSW-12		
Tipo	☐ funcional	☒ rendimiento	☐ usabilidad
Descripción	Será necesario que en caso de un fallo de hardware externo al sistema, este sea capaz de gestionarlo sin dejar de dar servicio. Los fallos considerados serán: - Pérdida de la conexión a Internet. - Error de acceso a BD.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Nuevos tests			
identificador	RSW-13		
Tipo	☐ funcional	☐ rendimiento	☒ usabilidad
Descripción	La descarga de nuevos tests será visible en la IU mediante el empleo de un contador.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Lenguaje sencillo			
identificador	RSW-14		
Tipo	☐ funcional	☐ rendimiento	☒ usabilidad
Descripción	El sistema tendrá un lenguaje sencillo para hacerlo cercano a los usuarios.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Controles propios			
identificador	RSW-15		
Tipo	☐ funcional	☐ rendimiento	☒ usabilidad
Descripción	El sistema adaptará los controles de Android, para darle un aspecto único a la interfaz y mejoren su visibilidad, con una buena representación de sus efectos.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Recuperación de errores			
identificador	RSW-16		
Tipo	☐ funcional	☐ rendimiento	☒ usabilidad
Descripción	El sistema adaptará los controles de Android, para darle un aspecto único a la interfaz.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Configuraciones de pantalla			
identificador	RSW-17		
Tipo	☐ funcional	☐ rendimiento	☒ usabilidad
Descripción	La aplicación se mostrará correctamente en el dispositivo móvil en posición vertical y horizontal (modo landscape)		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

7.1.3 Requisitos de restricción

Los requisitos de restricción son limitaciones impuestas sobre la manera en que el problema es resuelto y el objetivo es alcanzado. Dichos requisitos serán explicados de forma breve mediante tablas. Los campos para definir dichos requisitos de dividen en:

Título: Breve descripción del objetivo del requisito. No se indicará en la tabla como campo, sino que formará parte del título de la misma.

Identificador: Identificación unívoca del requisito. Debe seguir la nomenclatura RR-XX, dónde 'X' es un número entre 0 y 0, y se incrementarán en una unidad con cada nuevo requisito.

Descripción: Especifica el significado del requisito.

Estabilidad: Hace referencia a la sensibilidad del requisito a ser modificado. Toma los valores 'estable' o 'no estable'.

Prioridad: Define la prioridad con la que debe ser resuelto dicho requisito.

Puede tomar los valores de "Alta", "Media" y "Baja".

S.O. móvil			
identificador	RR-01		
Descripción	La versión mínima de Android para ejecutar la aplicación será Android 2.0		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

S.O. Servidor			
identificador	RR-02		
Descripción	El servidor correrá bajo Windows Xp.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Versión Java			
identificador	RR-03		
Descripción	La versión mínima de java necesaria para ejecutar el Servidor será java 1.6.0_17		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Permisos Android	
identificador	RR-04
Descripción	Al instalar la aplicación el usuario debe otorgar los permisos necesarios a la aplicación para que esta pueda funcionar de forma correcta en el teléfono. Estos permisos son: • Montar la unidad de almacenamiento externo. • Escribir en la unidad de almacenamiento externo. • Acceso a internet. • Consultar el estado de interfaz Wifi. • Consultar el estado de la interfaz de datos 3G.
Estabilidad	☒ Estable ☐ No estable
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Envío datos confidenciales	
identificador	RR-05
Descripción	No se permite el envío de datos confidenciales en claro a través de la red, por lo que la contraseña del usuario será cifrada.
Estabilidad	☒ Estable ☐ No estable
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

puertos servidor			
identificador	RR-06		
Descripción	El equipo servidor deberá tener habilitado y libre el puerto 8888 para la correcta comunicación con los dispositivos móviles.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Idioma de la interfaz			
identificador	RR-07		
Descripción	La interfaz mostrará sus contenidos en español. El idioma de los test no es responsabilidad del sistema.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

puertos servidor			
identificador	RR-07		
Descripción	Deberá existir conectividad Wifi o 3G para poder acceder a todas las funcionalidades de la aplicación.		
Estabilidad	☒ Estable	☐ No estable	
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

7.1.4 Casos de uso

En este apartado se utilizan los casos de uso como una simplificación gráfica de las funcionalidades del sistema para su mejor comprensión. Un caso de uso representa un uso típico del sistema, permitiendo dicha técnica capturar y definir los requisitos que se deben cumplir en una aplicación y la interacción que existe entre el usuario y el sistema. Es utilizada frecuentemente en el área de la Ingeniería del Software para mostrar al cliente de manera sencilla de qué será capaz su futuro sistema.

Sin embargo, aunque las posibilidades que nos ofrece esta técnica son importantes, no se va a profundizar en la misma, mostrando solo los casos de uso que resulten más ilustrativos al lector y que mejoren su comprensión global sobre la aplicación. Para ello se utilizará el estándar UML 2.0. Se detallará los pasos necesarios para cada escenario con un formato tabular descrito a continuación:

- Identificador: identifica de forma univoca un caso de uso siguiendo el formato de nombrado CU_X, donde X se corresponde a un número empezando en la unidad.
- Descripción: describe los pasos realizados por el usuario para la situación planteada.
- Pre-condiciones: condiciones que deben darse para la realización del caso de uso.

Identificador	CU_1: Identificación y ejecución de un test
Descripción	• El usuario se identifica en el sistema. • Elige la asignatura 'Redes de ordenadores' • Escoge un test • Completa cada pregunta del test. • Al terminar, elige corregir el test
Pre-condiciones	• Conexión a red Wifi o 3G • Tarjeta de memoria insertada y montada

Tabla 5: Primer caso de uso

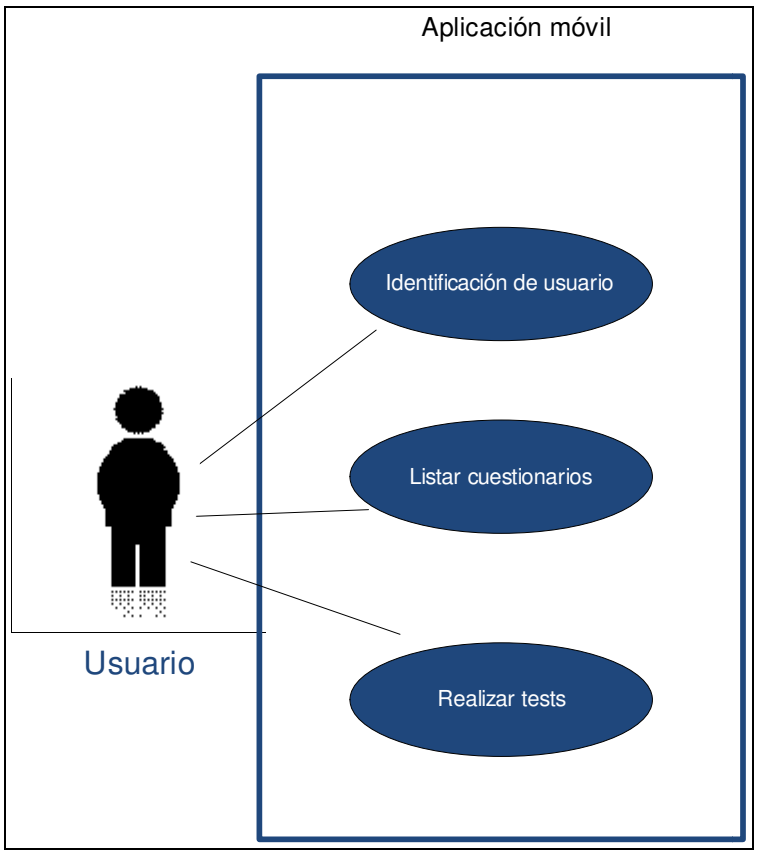


Ilustración 33: primer caso de uso

Identificador	CU_2: Ajustar opciones del test
Descripción	• El usuario navega hasta ajustes • Elije el modo de ejecución 'entrenamiento • Fija un límite de tiempo de 3 minutos para completar el test • Navega entre las asignaturas hasta encontrar el test deseado • Realiza el test, esta vez, cada pregunta se corrige al responderla
Pre-condiciones	• Tarjeta de memoria insertada y montada

Tabla 6: Segundo caso de uso

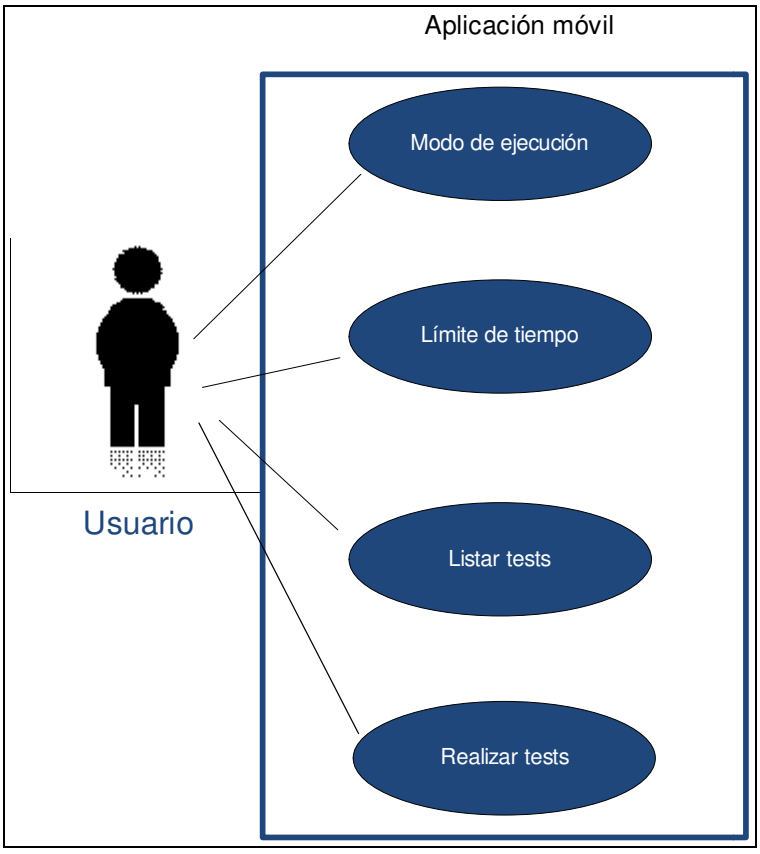


Ilustración 34: segundo caso de uso

Identificador	CU_3: Personalizar test
Descripción	• El usuario navega hasta ajustes • Elije el modo de ejecución 'entrenamiento • Fija un límite de tiempo de 3 minutos para completar el test • Navega entre las asignaturas hasta encontrar el test deseado • Realiza el test, esta vez, cada pregunta se corrige al responderla
Pre-condiciones	• Tarjeta de memoria insertada y montada

Tabla 7: tercer caso de uso

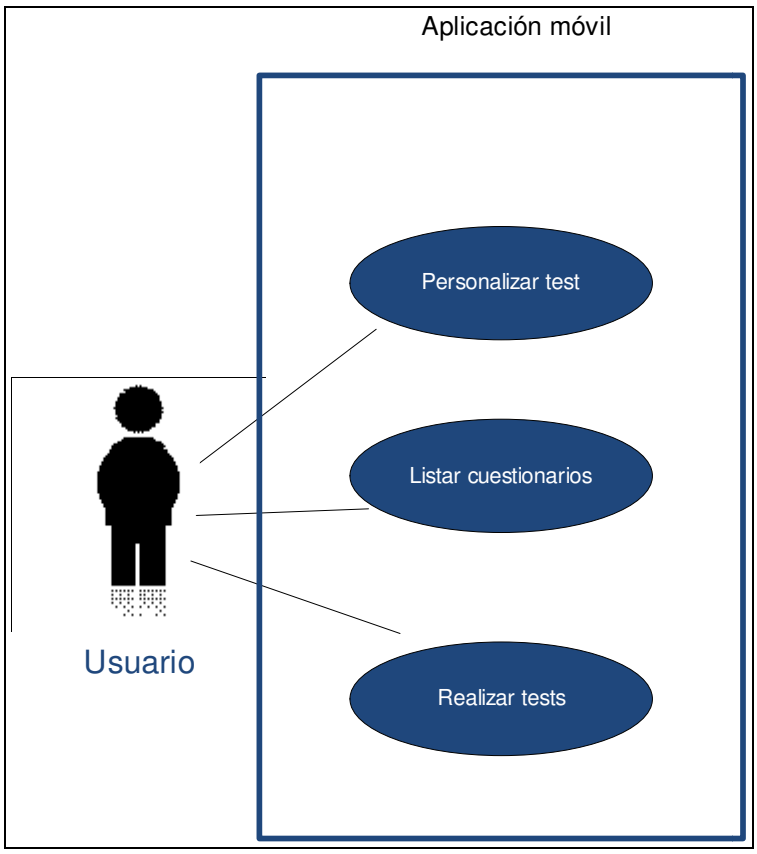
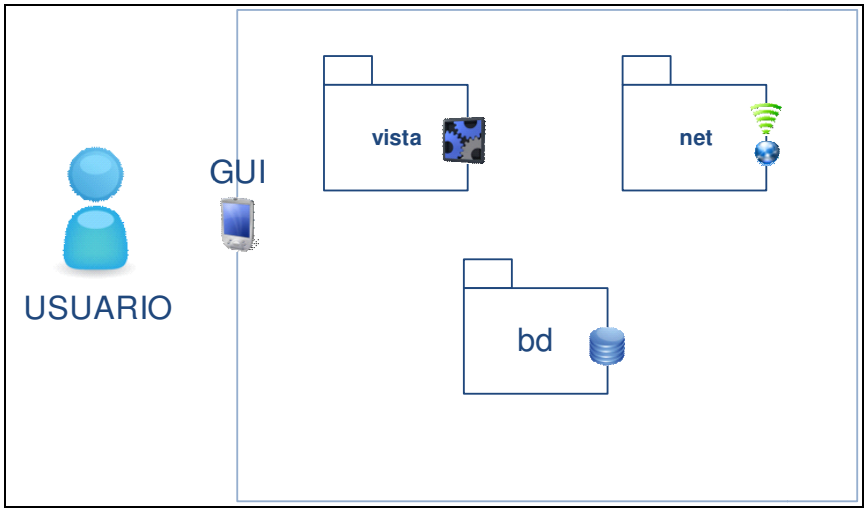


Ilustración 35: tercer caso de uso

7.2 Diseño

A la hora de diseñar la aplicación, a nivel lógico, podríamos dividirla en cada uno de los módulos que se muestran en la imagen a continuación:



El módulo **vista** alberga la lógica del programa: Las actividades y la relación entre ellas, así como las funcionalidades de cada una de ellas. En **net** se ha encapsulado la comunicación con el servidor y el protocolo de red. El módulo bd es responsable de controlar el acceso a la base de datos de la aplicación, así como de proporcionar una interfaz de los elementos de la base de datos para el resto de los módulos. Finalmente, habría que considerar aparte el diseño de la interfaz de usuario.

7.2.1 Paquete net

7.2.1.1 Protocolo de red

En el diseño de la aplicación se ha tenido en cuenta, tal y como se recoge en los requisitos de usuario, la comprobación de existencia de nuevos tests y su descarga en el dispositivo móvil. Aunque finalmente esta funcionalidad no ha sido incluida en la primera versión de la aplicación móvil, sí que se diseñó el protocolo de comunicación cliente-servidor correspondiente. Tras proporcionar una descripción general del mismo, se detallará el comportamiento de cliente y servidor en el protocolo. Se ha optado por emplear maquinas de estados finitos para su representación.

En la concepción del protocolo se ha tratado de reducir el número y tamaño de los mensajes intercambiados, habida cuenta de que, al ejecutarse el cliente en un dispositivo

móvil, es muy habitual que se produzca una pérdida de la conexión a Internet. Por ello, se tratará que la comunicación con el servidor se realice con brevedad.

7.2.1.1.1 Visión general

Al arrancar la aplicación, desde el cliente se intentará conectar al servidor, cuya URL y puerto se almacena en uno de los ficheros de recurso (ver detalles de implementación).

Si logra conectar, envía un mensaje de identificación, que incluye el NIA y la contraseña introducidos por el usuario en la aplicación móvil. Se acompañará esta información de un resumen de las asignaturas almacenadas en el dispositivo, incluyendo el código de la asignatura y su versión.

A continuación, desde el servidor se enviará el resultado de la identificación. En caso de haber identificado con éxito al cliente, acompañará el mensaje de las asignaturas (y sus tests) cuya versión sea superior a la indicada por el cliente el mensaje anterior.

En cuanto al formato de los mensajes, se ha decidido emplear XML, por tratarse del formato estándar para el intercambio de datos en servicios Web.

7.2.1.1.2 Versión de asignaturas

Para entender completamente el protocolo, es necesario describir con un mayor detalle la idea de 'versión de una asignatura' y 'versión de un test'.

Cada test almacenado en base de datos posee información relativa a su versión, esto es la fecha de su última modificación, en el siguiente formato:

yyyyMMddhhMMssmmm

Esto es:

Yyyy	Año
MM	Mes
Dd	Día
Hh	Hora (formato H24)
MM	Minutos
Ss	Segundos
Mmm	Milisegundos

Tabla 8: formato de fechas para la versión de las asignaturas

Así, sería un formato válido:

201104203059111

Cada test pertenece únicamente a una asignatura y la versión de una asignatura será la versión de su test más reciente.

Puede apreciarse en el siguiente diagrama:

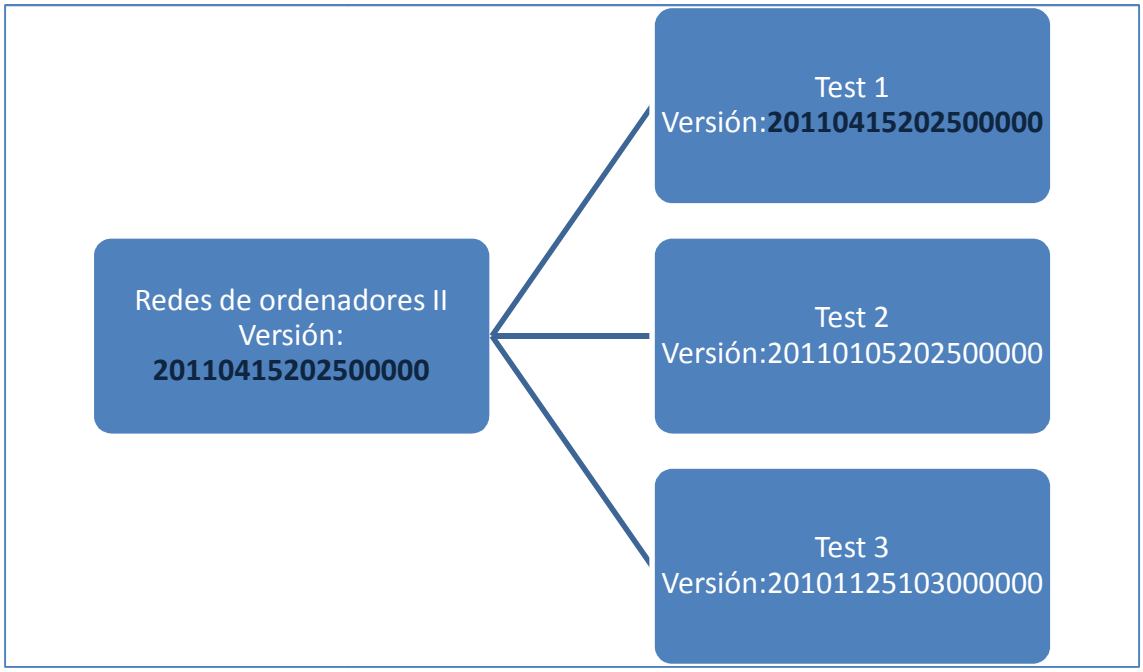


Ilustración 36: Relación entre versiones de asignaturas y cuestionarios

7.2.1.1.3 Especificación de mensajes

En este apartado describiremos la estructura de los mensajes del protocolo para el intercambio de información entre cliente y servidor.

7.2.1.1.4 Solicitud de actualizaciones

La solicitud se compone de: Un mensaje de actualización y N mensajes de versión de asignatura (de existir), tal y como describe a continuación:

Solicitud de actualizaciones

Tipo Mensaje	Cardinalidad
Mensaje de identificación	1
Mensaje de versión de asignatura	0…N

A continuación se describen los mensajes de identificación y versión de asignatura.

7.2.1.1.4.1.1 Identificación

Mensaje de identificación

NIA	El número de identificación de del alumno introducido desde la aplicación por el usuario
PW	Resultado de aplicar la función md5 a la contraseña introducida desde la aplicación por el usuario

7.2.1.1.4.1.2 Versión asignaturas

Para cada asignatura en la base de datos del cliente:

Mensaje de versión de asignatura

id-asignatura	El código que identifica de forma unívoca a una asignatura de una titulación y que es empleado por la universidad. Ej: Redes de ordenadores II, Ing. Superior Informática; Código = 13-11324
Versión-asignatura	La versión de la asignatura (versión de su test más reciente)

7.2.1.1.5 Respuesta a la solicitud

La respuesta se compone de: El resultado de la identificación y N mensajes de actualización de asignatura (de existir), tal y como describe a continuación:

Respuesta a la solicitud

Tipo Mensaje	Cardinalidad
Mensaje de resultado identificación	1
Mensaje de actualización de asignatura	0…N

Si falló la identificación del usuario no se incluyen mensajes de actualización de asignatura.

Si cliente y servidor poseen la misma versión de las asignaturas no se incluyen mensajes de actualización de asignatura.

7.2.1.1.5.1.1 Resultado identificación

Resultado de identificación

Éxito-id	Un código que indica el resultado de la identificación del usuario a partir de los datos proporcionados en el mensaje de solicitud de actualizaciones. **Éxito-id=0** El usuario se identificó con éxito. En tal caso, se incluirán mensajes de actualización en la respuesta a la solicitud, de haberlos. **Éxito-id=1** Se produjo un error en la identificación del usuario. No se incluyen mensajes de actualización en la respuesta.

7.2.1.1.5.1.2 Actualización asignatura

Mensaje de versión de asignatura

id-asignatura	El código que identifica de forma unívoca a una asignatura de una titulación y que es empleado por la universidad. Ej: Redes de ordenadores II, Ing. Superior Informática; Código = 13-11324
Versión-asignatura	La versión de la asignatura (versión de su test más reciente)
Listado-tests	Los nuevos tests de la asignatura, a actualizar en el cliente.

7.2.1.1.6 Autómatas de estados finitos

Se incluyen a continuación una descripción del protocolo –para cliente y el servidor - empleando autómatas finitos. Se proporcionará una descripción de cada estado.

7.2.1.1.6.1 Cliente

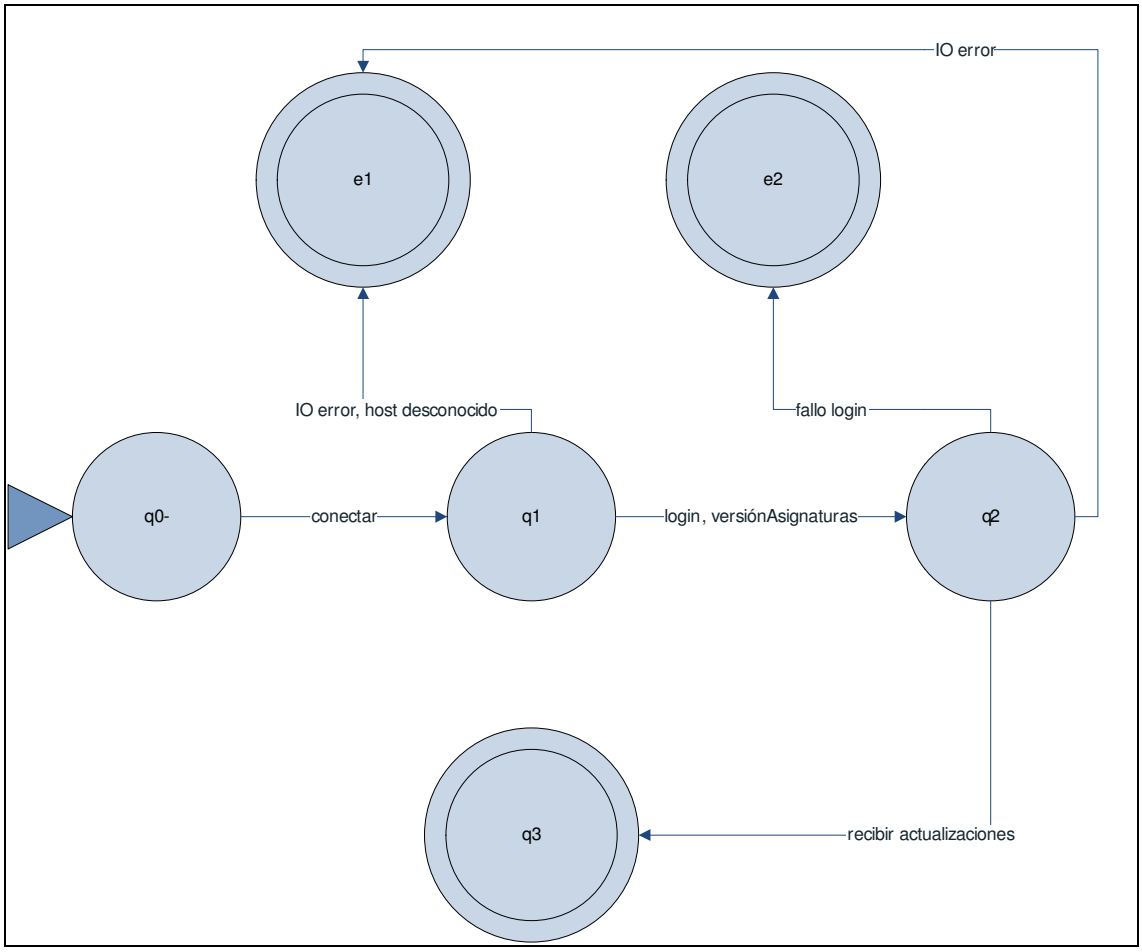


Ilustración 37: Diagrama de estados del cliente

- q0: **Estado inicial**. Tras intentar la conexión con el servidor, se transita a q1
- q1: De producirse un error de Entrada/Salida o no se encuentre el servidor, se transita al estado de error e1. Si el establecimiento de la conexión tiene éxito, se envía **el mensaje de solicitud de actualizaciones** y se transita a q2.
- q2: Si se recibe respuesta del servidor, se procesa la información **relativa al resultado de la identificación**. En caso de error, se transita al estado e2. De lo contrario, se procesa la parte del mensaje relacionada con la actualización de nuevos tests y se transita al estado final q3.
- q3: Estado final. Se llega a él si se produjo con **éxito la identificación** contra el servidor y se recibieron las **actualizaciones** – de haberlas – de las asignaturas.
- e1: Estado de error al que se transita en caso de producirse **un fallo de entrada/salida** al acceder a los lectores/escritores de los flujos de entrada/salida o si no se encontró el servidor.
- e2: Estado de error al que se transita si el **resultado de la identificación** del usuario no fue exitoso.

7.2.1.1.6.2 Servidor

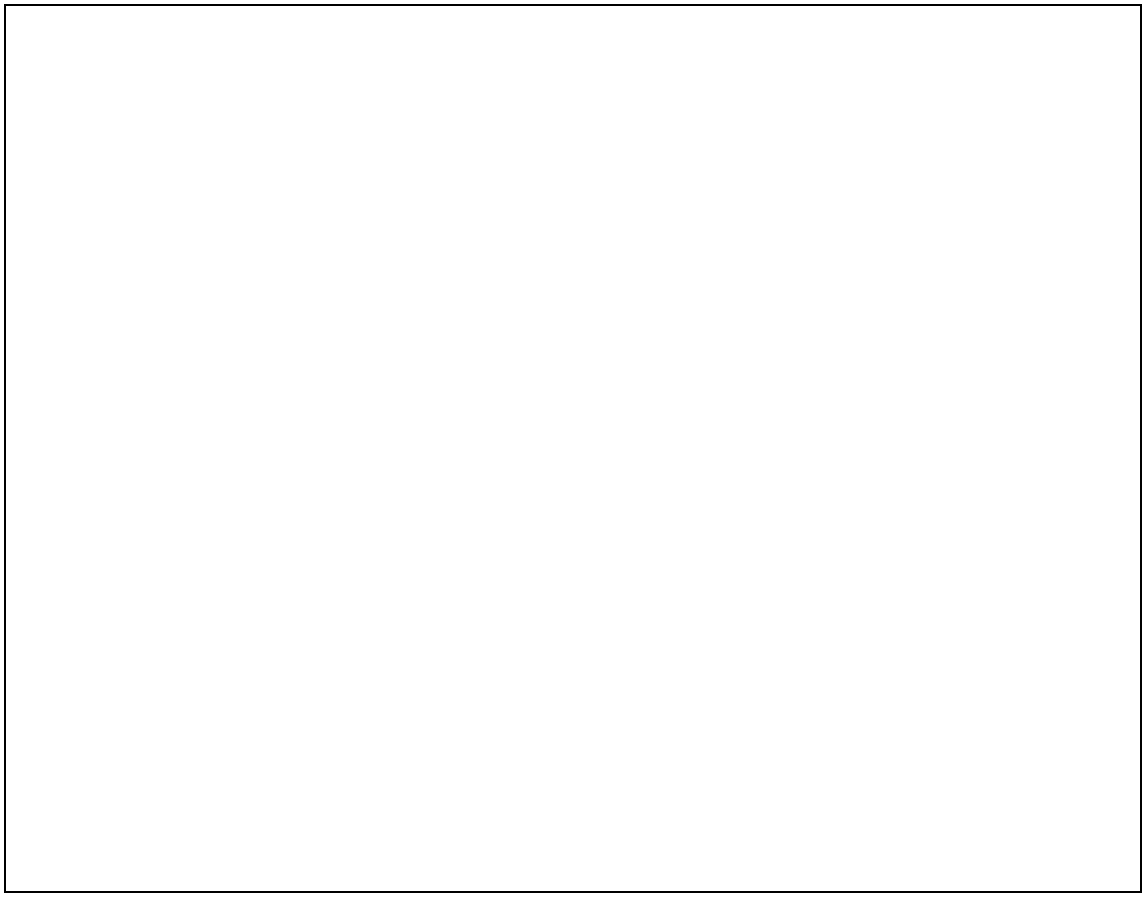


Ilustración 38: diagrama de estados del servidor

- q0: Estado inicial, el servidor está a la espera de conexiones. Cuando se recibe una petición, se transita a q1.
- q1: Se procesa la petición recibida: El servidor tratará de buscar el usuario en la base de datos, en caso de éxito, se envía el mensaje con la actualización de las asignaturas del usuario y se transita a q2. De lo contrario, se pasa al estado de error e1. Si se trata de una petición mal formada (o no es un mensaje propio del protocolo) se descarta el mensaje y se vuelve al estado inicial. De producirse un error de entrada/salida, se transita a q0.
- q2: Se cierra la conexión con el cliente y se vuelve al estado inicial.
- e1: Se envía al cliente el error en la identificación.

7.2.2 Paquete bd

7.2.2.1 Diseño de la base de datos

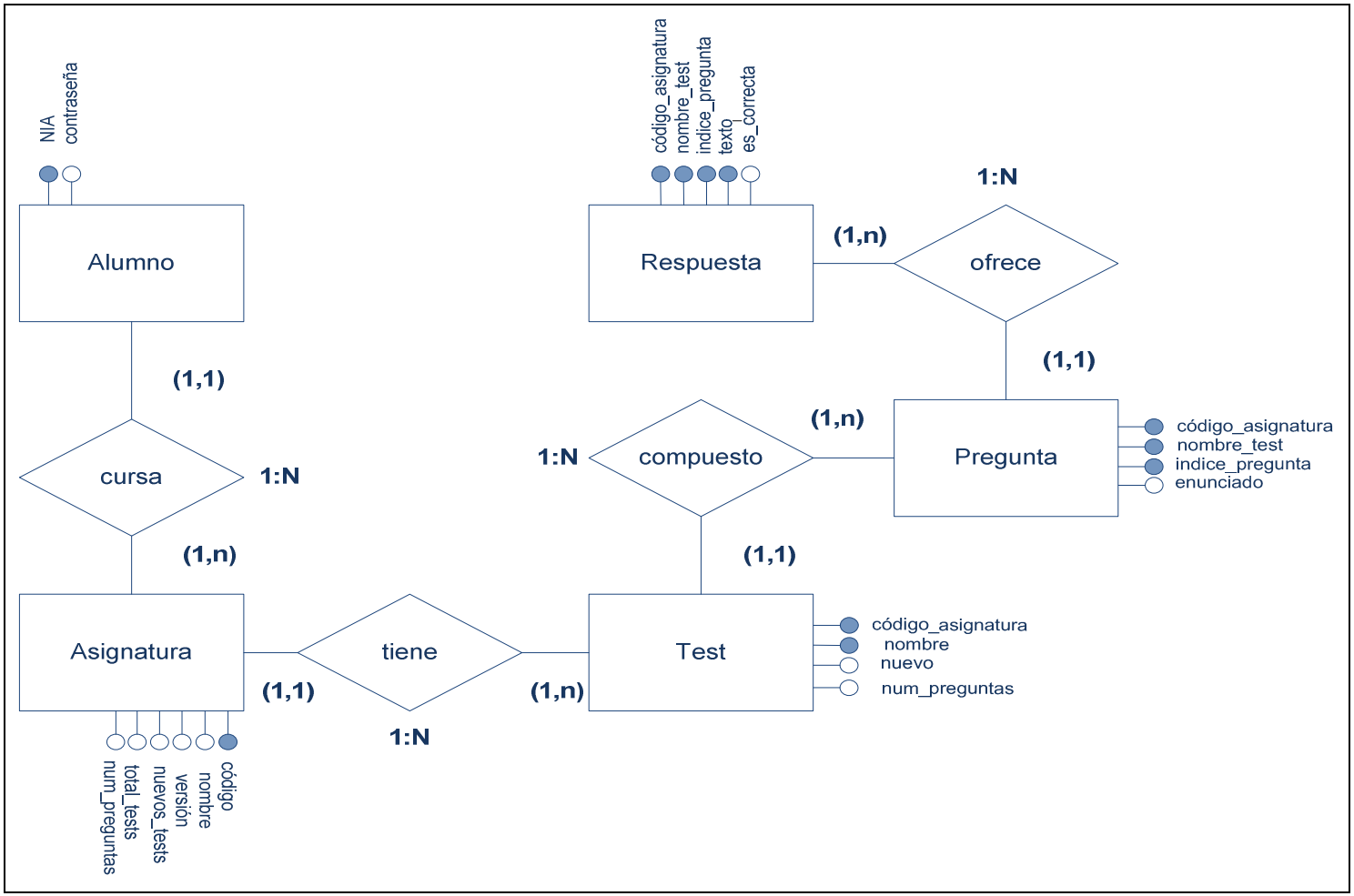


Ilustración 39: diseño de la base de datos

7.2.2.2 Disparadores

En el diseño de la base de datos se han incluido los siguientes disparadores:

- Tabla pregunta
  - Suma_pregunta: Incrementa el valor de la columna num_preguntas de las tabla Test y Asignatura para cada nueva pregunta insertada.
- Tabla Test
  - Incrementar_nuevos_tests:Cada vez que se inserta un nuevo test se ha de incrementar el contador del número de tests y el número de nuevos de la tabla asignatura
  - Decrementar_nuevos_tests: Para cada test marcado como nuevo, al borrarse, se decrementará el contador de nuevos de la asignatura a la que pertenezca el test.
  - Decrementar_nuevos_tests_2: Si el test no es nuevo, sólo se decrementa el contador del número de tests de la asignatura a la que pertenezca el test a borrar.
  - Marcar_test_nuevo: Pone a uno el valor de nuevo por cada inserción.

7.2.3 Paquete vista

A continuación se muestran el diagrama de clases simplificado del **paquete vista**, responsable de la lógica de la aplicación. En el apartado de implementación se entrará en detallarán con mayor profundidad algunos aspectos.

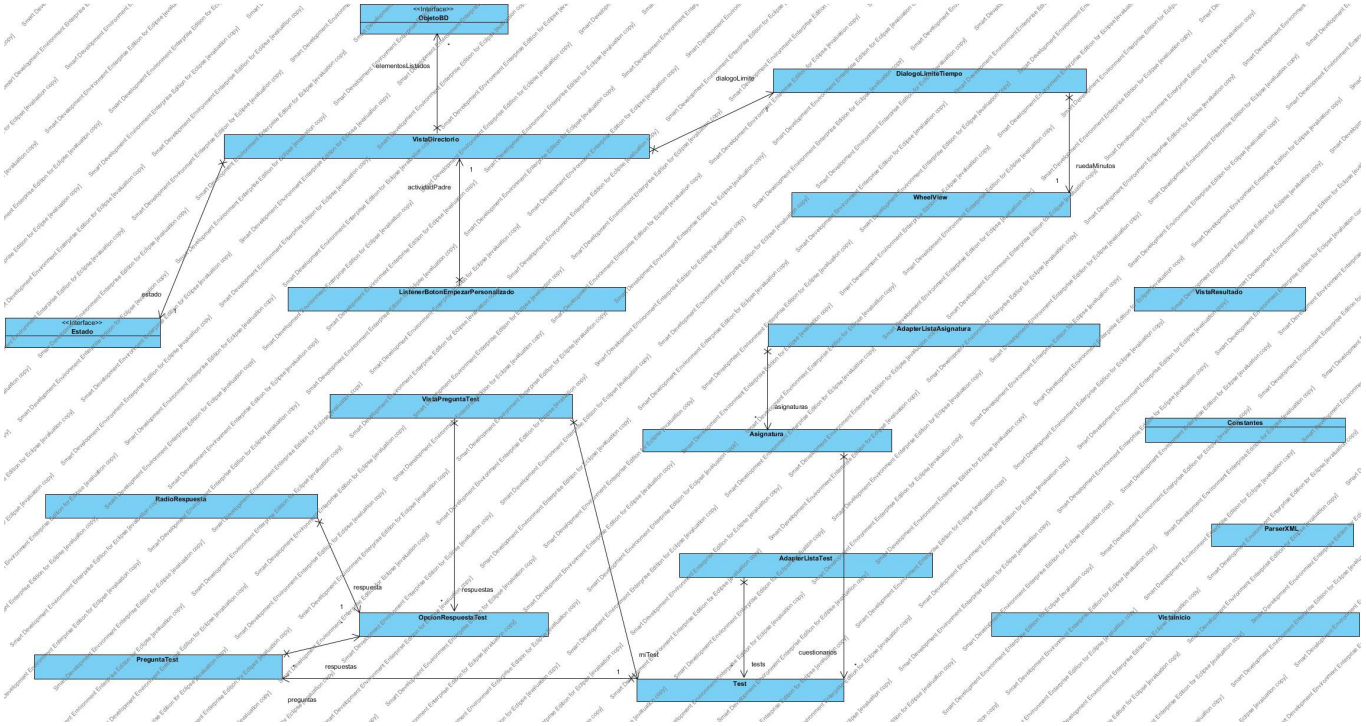


Ilustración 40: Diagrama de clases

7.2.4 Interfaz de usuario

El diseño de la interfaz de usuario se abordará mostrando cada una de las pantallas que componen la aplicación, haciendo una breve descripción de los controles que integra y de su función dentro de la aplicación.

En el proceso de desarrollo de la aplicación, el diseño de la interfaz gráfica es un aspecto que se ha tenido muy en cuenta y en el que se ha trabajado con especial dedicación.

Se ha procurado personalizar la mayoría de los controles del programa (listViews, botones, cuadros de texto, paneles tabulados, spinners, radio-botones…) a fin de dotar a la aplicación de una aspecto único que la haga reconocible entre el resto de aplicaciones similares. Otra de las razones para hacer hincapié en el diseño de la GUI es el de trabajar en un aspecto de la programación para Android que con frecuencia se ignora: La modificación de los controles de Android por defecto.

Las imágenes incluidas en esta sección no son capturas de pantalla de la aplicación, sino imágenes creadas con un editor, empleado el diseño de la interfaz. Las capturas se han incluido en la sección correspondiente a la GUI dentro del capítulo de detalles de implementación.

7.2.5 Identificación de usuarios

La primera pantalla de la aplicación. Muestra el logo de la misma, y dos campos de texto para introducir el NIA y la contraseña. A la derecha del campo de texto correspondiente a la contraseña se incluye un imageButton para completar la acción de identificación del usuario y enviar sus datos al servidor. A los campos de texto y al botón se les ha añadido sombra interior para dar sensación de profundidad y dotar de un aspecto 'clicable' a los controles

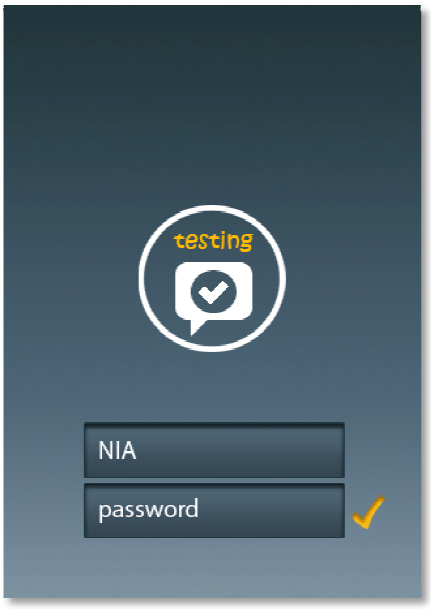


Ilustración 41: diseño de la pantalla de login

7.2.6 Pantalla principal

Tras completar el proceso de identificación – en caso contrario se muestra un mensaje de error y aplicación se mantiene en la pantalla de login – el sistema muestra la pantalla principal. En la parte superior se muestra el botón atrás – para salir de la aplicación o navegar entre listas – y los tipos de tests disponibles: descargados y personalizados. La opción seleccionada se muestra con un color más vivo.

El resto de la pantalla es un tabView personalizado. La pestaña seleccionada se muestra 'hundida' incluyendo sombra interior en el la opción, para dotar del aspecto clicable al control. El espacio restante se ocupa en mostrar la lista de test o asignaturas o los ajustes de la aplicación (límite de tiempo y modo de ejecución de los tests.

Para las asignaturas, se incluye el número de test disponibles y un contador de nuevos tests – el usuario no los ha realizado desde que se descargaron desde el servidor.

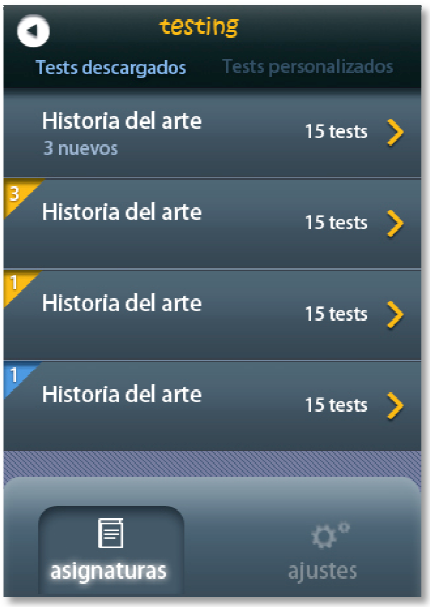


Ilustración 42: diseño de la pantalla principal

7.2.7 Pregunta de test (modo examen)

Tras escoger un test para la auto-evaluación, ésta será la pantalla que se muestre. Incluye, principalmente, el texto de la pregunta y las opciones de respuesta. Se han incluido en la pantalla controles para navegar entre preguntas anterior y siguiente (esquinas inferior derecha e izquierda) y la opción de regresar a la pantalla anterior, cancelando el test actual (esquina superior izquierda). Además, se informa del total de preguntas del test y del número de pregunta mostrada (abajo, en el centro). En el caso de elegir entre los ajustes de test un límite de tiempo, este se incluirá en la esquina superior izquierda, acompañado de la imagen de un cronómetro.

Para representar las respuestas se ha optado por radio botones, cuyo aspecto se ha personalizado de forma que ocupen todo el ancho de la pantalla, facilitando su selección – pues basta con hacer clic en el fondo, en lugar del clic sobre el texto o el radio-botón por defecto. Las respuestas modifican su aspecto al hacer clic sobre ellas (con el objetivo de aumentar el feedback) y al marcarlas, tras liberar la pulsación sobre ella.

La versión final incluye un botón con el texto 'corregir' que se puede observar en los detalles de implementación de la interfaz gráfica.

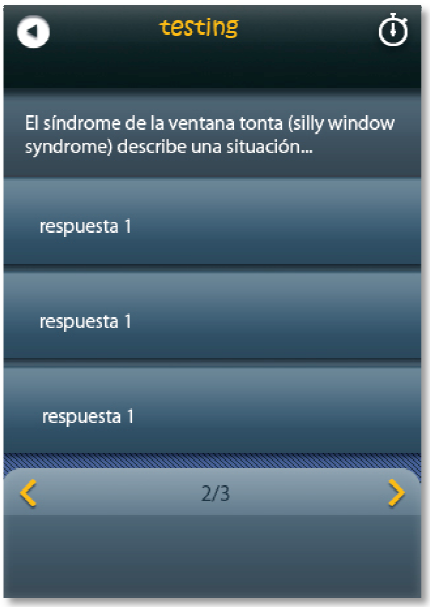


Ilustración 43: diseño de la vista de pregunta

7.2.8 Pregunta de test (modo entrenamiento)

Es similar a la anterior, con la salvedad de la ausencia de controles de navegación entre preguntas y del botón para corregir. No tiene sentido incluirlos en este diseño, pues en el modo entrenamiento, la respuesta correcta de cada pregunta se muestra de forma individual, al seleccionar una opción.

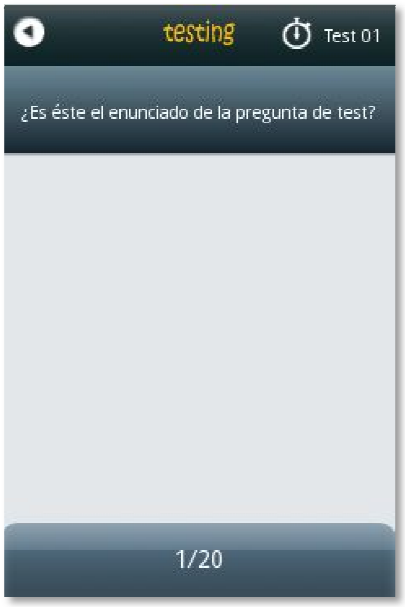


Ilustración 44: Diseño de la vista de preguntas (modo entrenamiento)

7.2.9 Resultado del test

Tras responder a la última pregunta del test, se muestran los resultados obtenidos en la pantalla de la imagen. Se incluye la opción de salir mediante el control situado en la parte superior izquierda de la ventana – común a toda la aplicación. Y la opción de repetir el test.

Se muestra la puntuación obtenida de forma numérica y gráfica. La imagen del birrete sólo se muestra cuando se logra la puntuación máxima en el test.

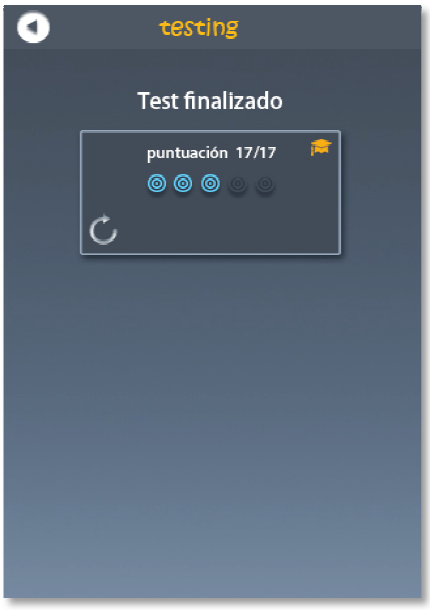
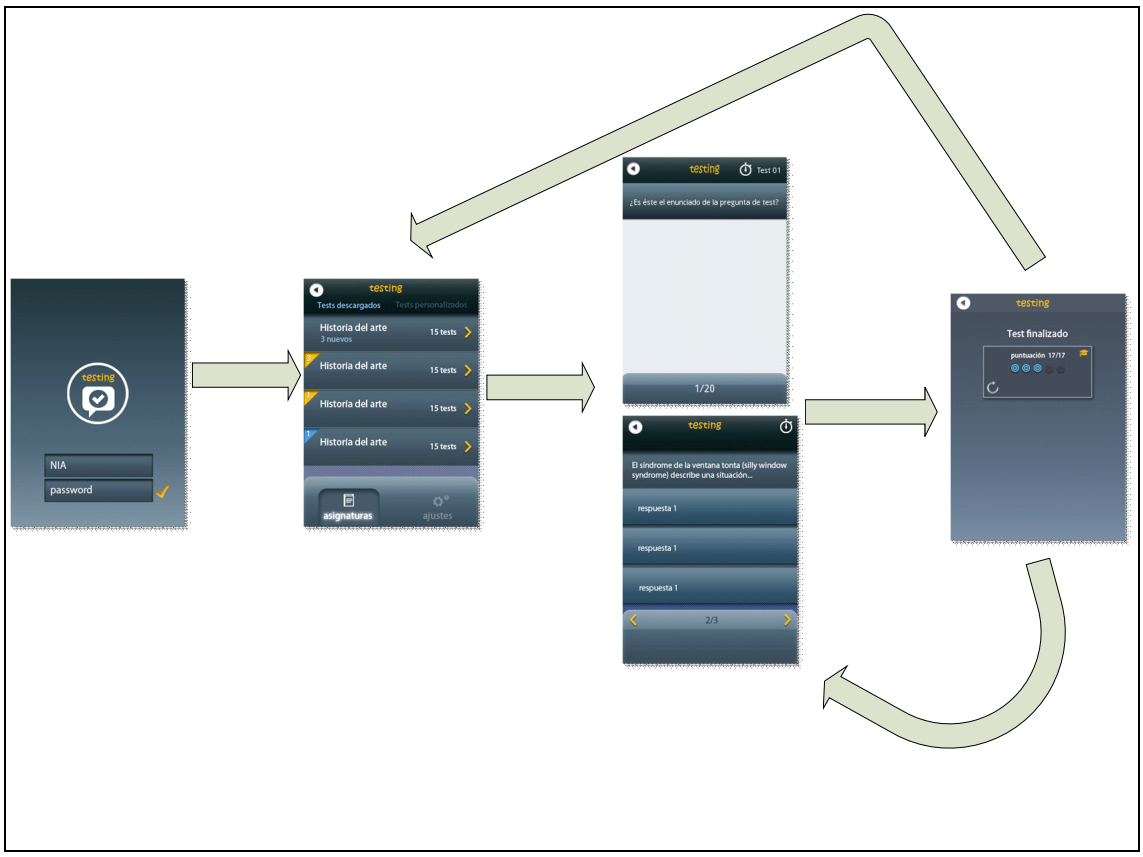


Ilustración 45: diseño de la vista de final de test

7.2.10 Diagrama de navegación

En este apartado se presenta el mapa de navegación del sitio, que detalla el orden en que se presentan las pantallas en la aplicación, y la flexibilidad para moverse entre ellas.



El punto de entrada a la aplicación será la pantalla de login. Tras completar el proceso de identificación, pasamos a la pantalla principal. Desde ahí podremos acceder a los cuestionarios. La siguiente pantalla mostrada¸la vista de los cuestionarios, estará relacionada con el modo de ejecución escogido. Tras completar el test se llega a la pantalla de resultado, que permite al usuario volver a la pantalla principal o repetir el cuestionario.

7.3 Detalles de implementación

7.3.1 Protocolo de comunicación

7.3.1.1 Configuración

Para almacenar los parámetros relativos a la comunicación en el cliente, se creó el fichero de recurso **conexión.xml**, a modo de **fichero de configuración**, que almacena los siguientes valores:

- Url: URL del servidor que recibirá conexiones de los clientes.
- Puerto: Puerto del servidor al que se conectará el cliente.
- Tiempo de espera: Tiempo (en milisegundos) que el cliente esperará que la petición de conexión sea aceptada por parte del servidor.

A continuación, se muestra el contenido del fichero:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="conexion_url_servidor">mdegayon.dyndns.org</string>
    <string name="conexion_puerto_servidor">8888</string>
    <string name="conexion_timeout">5000</string>
</resources>
```

De esta manera, se hace más **fácil adaptar el cliente** a un cambio en alguno de los parámetros anteriores.

7.3.1.2 Servidor de prueba

El servidor de prueba se ha suplido con el mismo equipo de desarrollo. Esto presentaba un problema, dado que el mentado equipo posee una **IP dinámica**. Para solventarlo, se ha optado por la solución DynDNS updater [2], aplicación de la empresa Dynamic Network Services, Inc. Dedicada a ofrecer resolución de nombres DNS en direcciones IP dinámicas.

Se registró la URL `mdegayon.dyndns.org` en el sitio web de DynDNS y se instaló en el equipo de desarrollo la aplicación **DynDNS updater**.

El servidor implementa una versión simplificada del protocolo de comunicación, de forma que, ante una identificación de usuario, una respuesta positiva si encontró el usuario o una respuesta negativa en caso contrario. Si el resultado de la identificación fue positivo, se envía a continuación el fichero XML con las asignaturas para el alumno logueado con éxito.

7.3.1.3 Esquemas

A continuación se incluyen los esquemas que definen los mensajes del protocolo de comunicación:

- Alumno.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/Alumno"
    xmlns:tns="http://xml.netbeans.org/schema/Alumno"
    elementFormDefault="qualified">
    <xsd:element name="nia" default="100033447"/>
    <xsd:element name="pw" default="miguel"/>
</xsd:schema>
```

- Asignatura.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/asignatura"
    xmlns:tns="http://xml.netbeans.org/schema/asignatura"
    xmlns:tst="http://xml.netbeans.org/schema/test"
    elementFormDefault="qualified">
    <xsd:import namespace="http://xml.netbeans.org/schema/test"
schemaLocation="test.xsd"/>



       <xsd:complexType name="tipoAsignatura">
            <xsd:sequence>
                <xsd:element name="codigo" type="xsd:string"
minOccurs="1" maxOccurs="1"/>
                <xsd:element name="nombre" type="xsd:string"
minOccurs="1" maxOccurs="1"/>
                <xsd:element name="test" type="tst:tipoCuestionario"
maxOccurs="unbounded" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    <xsd:element name="listadoAsignaturas">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="asignatura"
type="tns:tipoAsignatura" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="idAlumno" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

- Peticion.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/pfc33447/peticion"
    xmlns:tns="http://xml.netbeans.org/schema/pfc33447/peticion"
    elementFormDefault="qualified">
    <xsd:complexType name="version-asignatura">
        <xsd:sequence>
            <xsd:element name="id-asignatura"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

- Test.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/test"
    xmlns:tns="http://xml.netbeans.org/schema/test"
    elementFormDefault="qualified">

    <xsd:complexType name="pregunta">
        <xsd:sequence>
            <xsd:element name="enunciado" type="xsd:string"
default="Enunciado de la pregunta"/>
            <xsd:element name="verdadera" type="xsd:string"
maxOccurs="1" minOccurs="1" default="respuesta correcta"/>
            <xsd:element name="falsa" type="xsd:string" maxOccurs="10"
minOccurs="1" default="respuesta equivocada"/>
        </xsd:sequence>
    </xsd:complexType>


    <xsd:complexType name="tipoCuestionario">
        <xsd:sequence>
            <xsd:element name="nombre" type="xsd:string" minOccurs="1"
maxOccurs="1" default="Test 01"/>
            <xsd:element name="tema" type="xsd:string" minOccurs="1"
maxOccurs="1" default="Tema 01"/>
            <xsd:element name="preguntaCuestionario"
type="tns:pregunta" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="cuestionario" type="tns:tipoCuestionario"/>
</xsd:schema>
```

7.3.2 Base de datos

7.3.2.1 Sqlite Expert Personal 3

Antes de incluir la base de datos en la aplicación, ésta se ha desarrollado empleando la herramienta de gestión de bases de datos SQLITE **'SQLlite Expert Personal' [8]**, que crear y manipular tablas, triggers y el resto de los elementos de las bases de datos SQLITE. La siguiente imagen muestra una captura de la ventana de la herramienta:

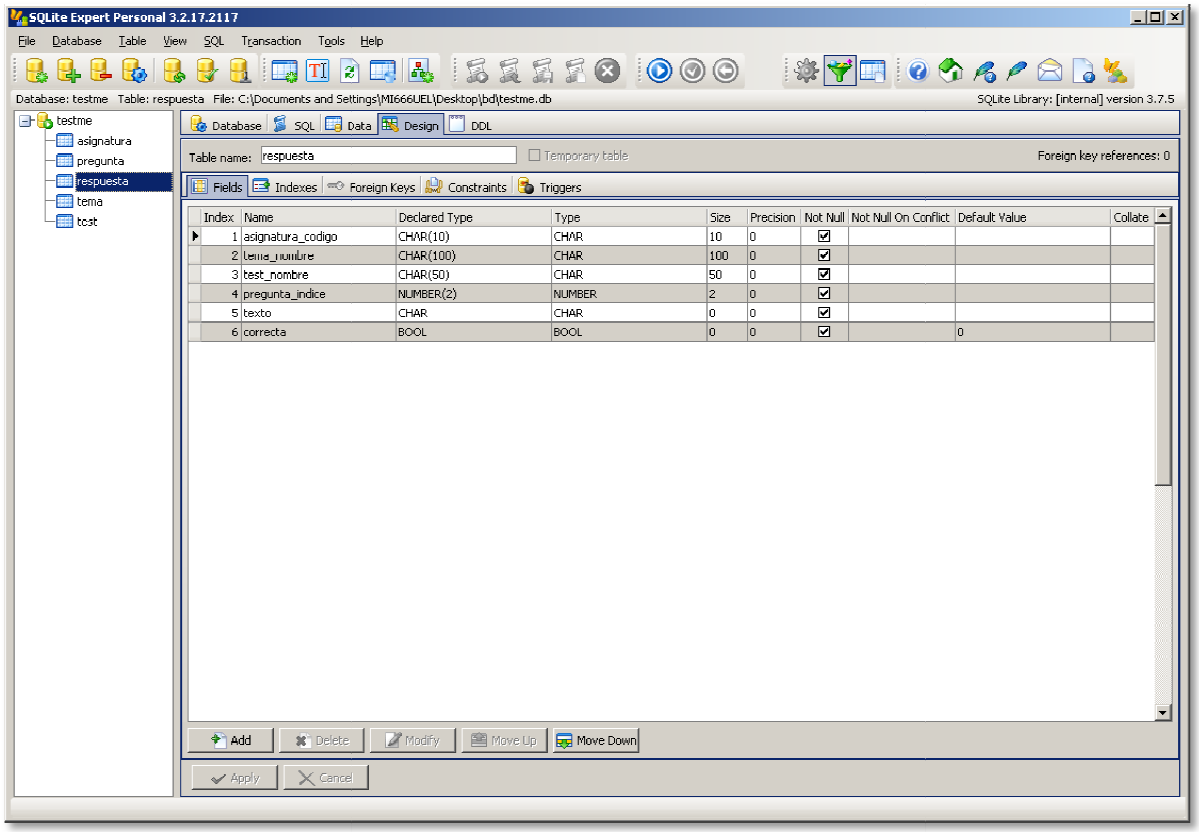


Ilustración 46: interfaz de SQLite Expert Personal

Tras crear las tablas la interfaz que facilita esta herramienta, se incluía su definición en la aplicación móvil. La declaración era fácilmente recuperable desde la herramienta, cómo muestra la figura:

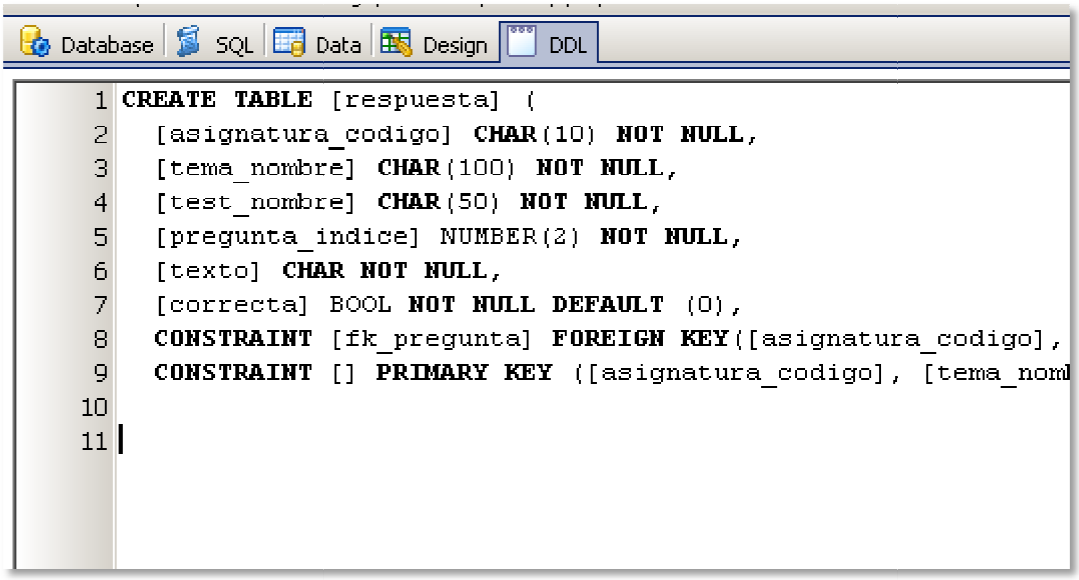


Ilustración 47: definición de tabla en SQLite Expert Personal

La creación de triggers es análoga a la de las tablas. Sólo hay que reseñar una pequeña diferencia – más bien, un añadido – y es que, antes de incorporarlos a la aplicación móvil, los disparadores, además de creados, eran probados desde la base de datos elaborada mediante SQLite Expert. De esta forma, se aseguraba que los disparadores ya funcionaban al incluirlos en el sistema.

7.3.2.2 Definición de la base de datos en la aplicación

Para almacenar los scripts de definición de las tablas y los disparadores, se creó el fichero de recurso **sqlite.xml**, dónde cada comando de definición de datos se almacena en un recurso de tipo String (tablas y disparadores).

De esta forma, se simplifica la modificación de la base de datos. Si se quiere incluir un nuevo disparador, basta con incluirlo en el fichero y apenas será necesario modificar el código de la aplicación. La siguiente imagen muestra la declaración de las tablas:

```
<!--  SCRIPTS DE CREACION DE TABLAS  -->

  <string name="crear_tabla_asignatura">
    CREATE TABLE [asignatura] (
      [codigo] CHAR(10) NOT NULL,
      [nombre] CHAR(100) NOT NULL,
      [nuevos_tests] NUMBER(2) DEFAULT (0),
      [total_tests] NUMBER(4) DEFAULT (0),
      [numero_preguntas] NUMBER DEFAULT (0),
      CONSTRAINT [sqlite_autoindex_asignatura_1] PRIMARY KEY ([codigo] ASC));
  </string>

  <string name="crear_tabla_tema">
    CREATE TABLE [tema] (
      [nombre] CHAR(100) NOT NULL,
      [asignatura_codigo] CHAR(10) NOT NULL CONSTRAINT [fk_asignatura]
      REFERENCES [asignatura]([codigo]),
      CONSTRAINT [] PRIMARY KEY ([asignatura_codigo], [nombre]));
  </string>

  <string name="crear_tabla_test">
    CREATE TABLE [test] (
      [nombre] CHAR(50) NOT NULL,
      [tema_nombre] CHAR(100) NOT NULL,
      [asignatura_codigo] CHAR(10) NOT NULL,
      [nuevo] BOOL DEFAULT (1),
      [numero_preguntas] NUMBER DEFAULT (0),
      CONSTRAINT [fk_tema] FOREIGN KEY([asignatura_codigo], [tema_nombre])
        REFERENCES [tema]([asignatura_codigo], [nombre]),
      CONSTRAINT [sqlite_autoindex_test_1] PRIMARY KEY
        ([asignatura_codigo], [tema_nombre], [nombre]));
  </string>
```

Ilustración 48: fichero de recurso con los scripts de creación de tablas

A continuación, el comienzo de la declaración de los disparadores en el fichero **sqlite.xml**:

```
<!--  SCRIPTS DE CREACION DE TRIGGERS  -->

  <string name="incrementar_nuevos_tests">

    CREATE TRIGGER [incremenentar_nuevos_tests]
      AFTER INSERT
      ON test
      FOR EACH ROW
      BEGIN

      update asignatura set nuevos_tests = nuevos_tests + 1 where
        codigo=new.asignatura_codigo;
      update asignatura set total_tests = total_tests + 1 where codigo=new.asignatura_codigo;

      END;
  </string>

  <string name="decrementar_nuevos_tests">
    CREATE TRIGGER [decrementar_nuevos_tests]
    AFTER UPDATE OF [nuevo]
    ON test
    FOR EACH ROW
    WHEN new.nuevo=0
    BEGIN

    update asignatura set nuevos_tests = nuevos_tests - 1 where
      codigo=new.asignatura_codigo;

    END;
  </string>
```

Ilustración 49: scripts de creación de triggers en el fichero de recurso

7.3.3 Interfaz de usuario

7.3.3.1 Versiones previas

Antes de dar con el diseño definitivo de la interfaz, tal y como se ve en la sección de diseño, las pantallas de la aplicación mostraron un aspecto, en algunos casos muy diferente. En ocasiones, el tamaño de los controles era demasiado grande y se hacía necesario cambiar su distribución en la interfaz, en otros no presentaban una representación adecuada de sus efectos. Vamos a hacer un repaso de algunas de esas pantallas descartadas cuyo diseño también forma parte del desarrollo y que ha permitido identificar y corregir ciertos errores.

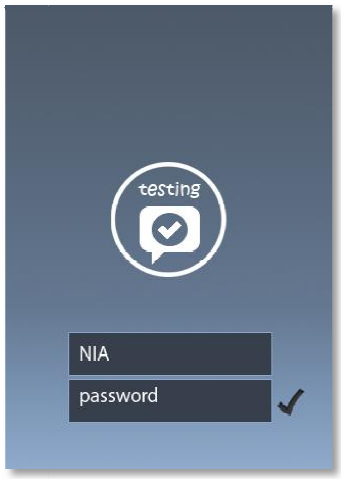
7.3.3.1.1 Identificación

La primera de ellas corresponde a la primera versión de la aplicación, con una paleta de colores completamente diferente.

La segunda la sustituyó poco después. La última versión es la que más se parece al diseño definitivo, que incluyó algo más de color en el logo y el botón de siguiente.

Ilustración 50: primera versión de la vista de login

Ilustración 51: Segunda versión de la vista de login



ración 52: una de las nas versiones de la vista de

7.3.3.1.2 Principal

El primer diseño únicamente permitía desde la pantalla principal listar los tests, elegir el modo de aplicación y un botón para retroceder/salir. Al resto de acciones se accedería mediante la tecla menú del teléfono. Esto se descartó para incluir los ajustes en la vista.

El segundo diseño es muy similar al definitivo, cambiando únicamente el aspecto del tabbedView y los elementos de la lista, de colores planos. En el diseño definitivo se incluyeron degradados que daban una sensación de elemento clicable a los controles.

las primeras
rincipal

Ilustración 53: una de las últimas versiones de la vista principal

7.3.3.1.3 Pregunta de test

El primer diseño presentaba numerosos problemas que debían ser corregidos. Entre ellos, el gran tamaño del bloque inferior dedicado a los controles, que dificultaba la visualización de la pregunta en el caso de que el enunciado de ésta o el texto de las respuestas fuera algo grande, pues requería de la barra de scroll vertical. Además, los controles de cancelar y corregir se encontraban demasiado cerca, y la ausencia de texto hacía difícil imaginar sus efectos.

La segunda de ellas posee un estilo más cercano al definitivo, sin embargo, presenta algunos de los problemas de su predecesora. En este caso, se optó por incluir los controles para **cancelar** y **corregir** en las opciones de la tecla **menú** del teléfono.

El último diseño es el más parecido al definitivo, sin embargo, el tamaño del botón 'corregir' seguía ocupando demasiado espacio. En la versión definitiva se optará por mover el número de pregunta actual a la izquierda del logo de la aplicación, ocupando su posición en la parte inferior un botón de 'corregir' de menor tamaño.

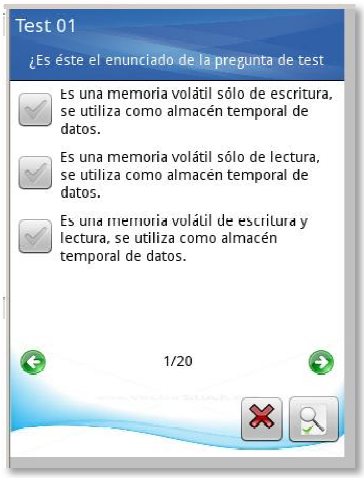
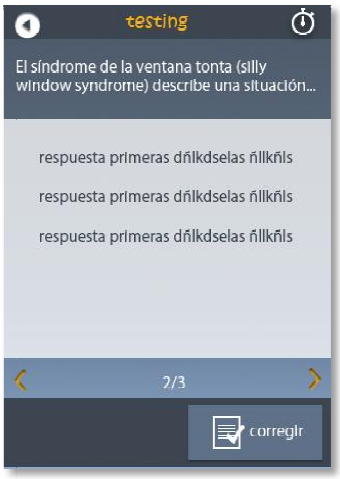


Ilustración 56: Primera versión de la vista de la pregunta de test

Ilustración 55: Revisión de la vista de pregunta de test



ustración 57: Una de las ltimas versiones de la vista de regunta de test

7.3.3.1.4 Resultado

El primer diseño era muy sencillo, incluía el número de fallos y aciertos y dos controles en la parte inferior que permitían volver a la pantalla principal o repetir el test.

El último es, en varios aspectos, similar al definitivo: incluye una representación gráfica del resultado, ImageButtons para volver al inicio y repetir el test y muestra el resultado en la parte superior de la vista, dentro de un cuadro.

versión de test

ión de

7.3.3.2 Ficheros de layout

Los ficheros XML que finalmente fueron incluidos en la aplicación, poseen un aspecto, en algún caso, ligeramente diferente al mostrado en el apartado de diseño. A fin de conocer el diseño definitivo de la interfaz de usuario, se incluyen a continuación los layouts de las actividades de la aplicación:

7.3.3.2.1 Identificación

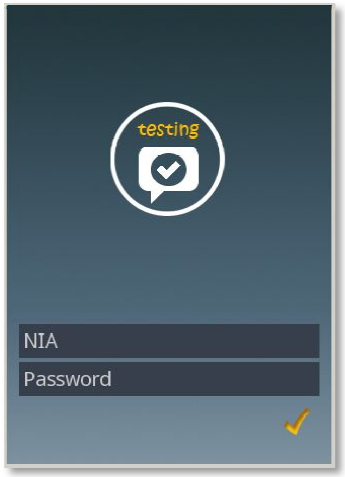


Ilustración 60: login.xml

7.3.3.2.2 Principal

Ilustración 61: principal.xml

7.3.3.2.3 Pregunta de test (entrenamiento)

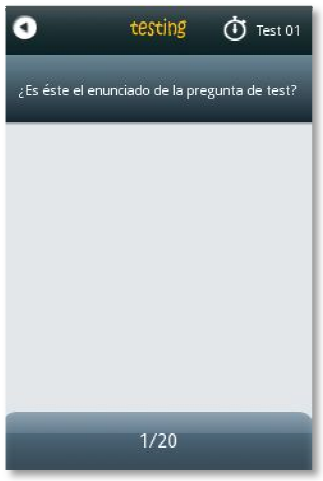


Ilustración 62: entrenamiento.xml

7.3.3.2.4 Pregunta de test (examen)

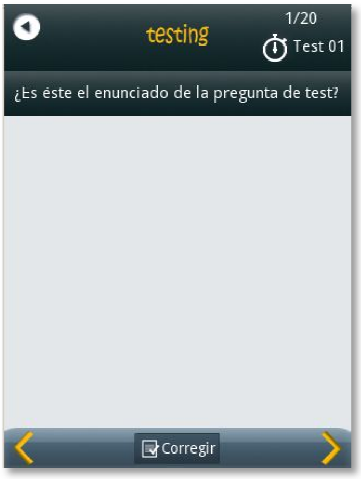


Ilustración 63: pregunta.xml

7.3.3.2.5 Resultado

Ilustración 64: resultado.xml

7.3.3.3 Diseño modular

Tal y como recomienda Android en su guía para desarrolladores [3], se ha decidido por hacer un diseño modular de los layouts, de forma que sea más fácil reutilizar y compartir su código. Opciones como los ajustes, los elementos de las listas o el aspecto de las respuestas se definen en sus propios layouts. Para incluirlos en otro, basta con emplear la etiqueta **<include>** en los ficheros XML. Sirva como ejemplo el diseño de la pantalla de ajustes, que se ha visto simplificado en gran medida, pues en el documento XML únicamente aparece un LinearLayout, que incorpora la etiqueta <include> por cada ajuste en la aplicación. A continuación, este ejemplo del uso de **include** en el código del proyecto:

```
<LinearLayout
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:id="@+id/layoutAjustes"
    android:orientation="vertical"
    android:visibility="invisible">

    <include
        android:id="@+id/modos"
        layout="@layout/ajustes_modo_ejecucion"/>

    <include
        android:id="@+id/limite"
        layout="@layout/ajustes_limite_tiempo"/>

</LinearLayout>
<FrameLayout
```

Ilustración 65: uso de include en los fichero de xml

Se muestran, seguidamente, los layouts implementados individualmente, de forma que, examinarlos junto a las imágenes incluidas en las secciones de **capturas de pantalla** y **ficheros de layout**, permita al lector comprender como se integra cada layout para formar la pantalla mostrada al usuario.

7.3.3.3.1 Ajuste de límite de tiempo

Ilustración 66: ajustes_limite_tiempo.xml

7.3.3.3.2 Ajuste de modo de ejecución

Ilustración 67: ajustes_modo_ejecucion.xml

7.3.3.3.3 Número de preguntas del test

Ilustración 68: personalizado_numero_preguntas.xml

7.3.3.3.4 Elemento de la lista de asignaturas para tests descargados

Ilustración 69: asignatura.xml

7.3.3.3.5 Elemento de la lista de asignaturas para tests personalizados

Ilustración 70: asignatura_personalizada.xml

7.3.3.3.6 Diseño del dialogo para escoger el límite de tiempo de la aplicación.

Ilustración 71: dialogo_tiempo.xml

7.3.3.3.7 Respuesta de test

Ilustración 72: respuesta.xml

7.3.3.3.8 Elemento de la lista de asignaturas

Ilustración 73: test.xml

7.4 Pruebas y evaluación de usabilidad

7.4.1 Evaluación de usabilidad

Al hablar de la usabilidades, nos referimos a la cualidad de un sistema respecto a:

- su facilidad de uso: múltiples formas de intercambiar información entre el usuario y el sistema
- su facilidad de aprendizaje para nuevos usuarios que garantizan interacción efectiva y máximas prestaciones
- y la satisfacción del usuario incluyendo el soporte al usuario para garantizar las metas (robustez)

Existen diferentes técnicas para medir la usabilidad de un sistema, que nos permitirán establecer si un sistema cumple las necesidades y expectativas del usuario y si, por lo tanto, éste se encuentra satisfecho. Con tal fin, se ha realizado el proceso de evaluación de la aplicación móvil, empleando la técnica conocida como 'Test de usuarios'. Se basa en la observación y análisis de un grupo de usuarios reales usando el sistema.

Se acordará con los usuarios, que no deben contar con más información de la necesaria sobre el sistema, de forma que actúen con libertad en su interacción con la aplicación, las tareas a realizar con el sistema.

7.4.1.1 Perfil de los evaluadores

Serán cinco los usuarios escogidos para llevar a cabo la prueba, el perfil de los evaluadores (el nombre que reciben los usuarios elegidos para interactuar con el sistema) es el siguiente:

Nombre	Número de evaluadores	Características
Usuario experto Android	1	Desarrollador software con conocimientos de informática y usabilidad. Usuario de Android
Usuario experto Otros S.O.	2	Desarrollador software con conocimientos de informática y usabilidad. Usuario de otros sistemas operativos móviles
Usuario básico Android	2	Usuario de Android. Conocimientos de informática a nivel de usuario.

Tabla 9: perfil de los evaluadores

7.4.1.2 Especificación de tareas

Se escogieron tres tareas para los evaluadores, que abarcan gran parte de la interacción del usuario con el sistema. Son las siguientes:

Nombre de tarea	Descripción
Realizar Test	Realizar el test 'Febrero 2006' de la asignatura 'Redes de ordenadores 2'.
Personalizar	Personalizar un test de la asignatura 'Ingeniería del software II' con 15 preguntas
Test de Entrenamiento	Realizar un test 'nuevo' en modo 'entrenamiento' con un tiempo límite de 5 minutos.

Tabla 10: tareas del test de usabilidad

7.4.1.3 Proceso de evaluación

Antes de comenzar la evaluación, se le hace saber al evaluador que el objetivo de la prueba es evaluar la calidad de uso de la aplicación, nunca la evaluación del participante. Si el participante comete algún fallo durante la prueba, no será culpa suya, sino del diseño.

Al comenzar la prueba, al evaluador se le introduce en la aplicación mediante el siguiente texto breve:

> TESTING es una aplicación móvil que permite realizar test de auto-evaluación para la universidad. Tu NIA es **100033447** y tu contraseña es **miguel**

El texto viene acompañado de la tabla con las tareas a realizar incluida en el apartado anterior.

El objetivo de facilitar el texto anterior es el de evitar cometer el error de explicar al usuario la aplicación a evaluar, ya que de lo que se trata es de comprobar el grado en que resulta auto-explicativa, clara y fácil de comprender.

Durante la evaluación, el usuario nos hace saber expresando qué problemas encuentra, qué no entiende o qué cree que significa cada elemento. Es importante no responder ni ayudar al usuario en la consecución de tareas.

7.4.1.4 Resultados y cambios en la interfaz

En esta sección analizaremos los cambios realizaros y haremos un resumen de los cambios que, como resultado de la evaluación, se han realizado en la interfaz de usuario.

7.4.1.4.1 Resultados

La siguiente tabla recoge los tiempos y la dificultad que para el usuario supuso cada tarea. El grado de dificultad nos fue facilitado por el usuario, dentro de una escala con los siguientes valores:

- muy fácil
- fácil
- media
- difícil
- muy difícil

El tiempo aparece medido en minutos. En casos en los que figura N/A (no aplica), el usuario no completó la tarea.

Evaluador	Realizar Test		Personalizar Test		Test de entrenamiento	
	tiempo	dificultad	tiempo	dificultad	tiempo	dificultad
Usuario Experto Android 1	2	Fácil	2	Media	4	Media
Usuario Experto Otros S.O 1	1	Fácil	2	Media	N/A	Muy difícil
Usuario Experto Otros S.O 2	3	Media	2	Fácil	4	Difícil
Usuario Básico Android 1	2	Media	2	Media	N/A	Difícil
Usuario Básico Android 2	2	Fácil	3	Difícil	N/A	Muy difícil

Tabla 11: resultados del test

7.4.1.4.2 Cambios

A partir de los problemas encontrados por los usuarios en la realización de las tareas, se he decidido incluir las siguientes modificaciones en la aplicación:

- **Escoger tamaño del texto de las respuestas**: A dos de los usuarios les parecía demasiado pequeño y les resultó difícil su lectura. Los otros tres no tuvieron problema ni hicieron mención al respecto, por lo que se ha optado por permitir al usuario escoger el tamaño del texto.

- **Modificar aspecto de modo de test:** Un usuario encontró confuso el control escogido para seleccionar el modo de ejecución del test, indicando como adecuado el escogido para elegir el límite de tiempo. Éste será el aspecto que mostrará el control, dándo de este modo, un aspecto uniforme a la pantalla de ajustes.

- **Mover el tipo de test a pestaña:** El mayor problema de la interfaz parece encontrarse en la opción de personalizar test, que ha supuesto un problema para cuatro de los cinco usuarios. No entendían, en primer lugar, el significado de los textos 'tests descargados' y 'personalizar test'. Uno de los evaluadores ni siquiera lo reconoció como un control 'clicable'. Además en dos casos no parecían entender los efectos que tendría sobre la aplicación su selección. Por ello se incluirá una tercera pestaña en el tabbed pane inferior, con el texto 'tipo de test', que permitirá escoger – mediante un control 'spinner '– el tipo de test. Esto vendrá acompañado de una explicación de cada tipo.

- **Incluir Ver por asignaturas/ver todos:** En la pantalla de la lista de asignaturas, antes de elegir el test, dos evaluadores no supieron qué se listaba: No sabían si se trataba de un test. Hicieron 'clic' para comprobar que efectos tendría, y al mostrar el listado de test correspondientes a la asignatura clicada, uno de ellos expresó dudas. Se incluirá, como paso previo, la opción de 'mostrar todos los tests' y 'mostrar por asignaturas'.

8 PLANIFICACIÓN Y PRESUPUESTO

En este capítulo se va a detallar la planificación seguida para la elaboración del proyecto. Se ha de tener en cuenta que la dedicación a este proyecto no ha sido exclusiva, Pues ha sido necesario compatibilizarlo con la actividad profesional, que impedía trabajar en el desarrollo de jueves a viernes. Esto se ha tenido en cuenta en la planificación, fijando el siguiente horario:

- Lunes - miércoles de 09:00 a 14:00

Estimando 5 horas al día para la realización del proyecto. La siguiente tabla muestra las actividades que han integrado el desarrollo. Se muestra la fecha de inicio y finalización de cada una de ellas y la duración (en días) de cada actividad. Se incluyen los recursos empleados y las sub-tareas en las que se descompone cada una de las tareas.

8.1 Planificación

Para realizar la planificación del proyecto se ha utilizado un diagrama de Gantt que podemos observar a continuación.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombre de los recursos
Estudio de viabilidad	9 días	lunes, 29 de noviembre de 2010	miércoles, 01 de diciembre de 2010		
Análisis de sistemas operativos móviles	3 días	lunes, 29 de noviembre de 2010	martes, 30 de noviembre de 2010		Analista
Estudio de otras soluciones	3 días	martes, 30 de noviembre de 2010	miércoles, 01 de diciembre de 2010	2	Analista
Fin del estudio de viabilidad	0 días	miércoles, 01 de diciembre de 2010	miércoles, 01 de diciembre de 2010	3	
Documentación	12,5 días	miércoles, 01 de diciembre de 2010	lunes, 06 de diciembre de 2010	3	
Tutoriales on-line	3 días	miércoles, 01 de diciembre de 2010	lunes, 06 de diciembre de 2010		Analista
Documentación on-line	3 días	miércoles, 01 de diciembre de 2010	lunes, 06 de diciembre de 2010	6FF	Analista
Lectura de la Guía para Desarrolladores	3 días	miércoles, 01 de diciembre de 2010	lunes, 06 de diciembre de 2010	7FF	Analista
Fin de la fase de documentación	0 días	lunes, 06 de diciembre de 2010	lunes, 06 de diciembre de 2010	8	
Análisis	28,5 días	lunes, 06 de diciembre de 2010	miércoles, 15 de diciembre de 2010	8	
Propuesta del proyecto	1 día	lunes, 06 de diciembre de 2010	lunes, 06 de diciembre de 2010		Analista
Recogida de requisitos	12 días	martes, 07 de diciembre de 2010	miércoles, 15 de diciembre de 2010	11	Analista
Fin de la fase de análisis	0 días	miércoles, 15 de diciembre de 2010	miércoles, 15 de diciembre de 2010	12	
Diseño	80 días	miércoles, 15 de diciembre de 2010	miércoles, 12 de enero de 2011	12	
Diseño del protocolo de red	6 días	miércoles, 15 de diciembre de 2010	miércoles, 22 de diciembre de 2010		Diseñador

Diseño de la base de datos	6 días	miércoles, 22 de diciembre de 2010	martes, 28 de diciembre de 2010	15	Diseñador
Diseño de la interfaz gráfica	9 días	martes, 28 de diciembre de 2010	miércoles, 05 de enero de 2011	16	Diseñador
Diseño de la aplicación	9 días	miércoles, 05 de enero de 2011	miércoles, 12 de enero de 2011	17	Diseñador
Fin de la fase de diseño	0 días	miércoles, 12 de enero de 2011	miércoles, 12 de enero de 2011	18	
Implementación	99,5 días	miércoles, 12 de enero de 2011	miércoles, 16 de febrero de 2011	18	
Configuración del entorno de desarrollo	1 día	miércoles, 12 de enero de 2011	lunes, 17 de enero de 2011		Diseñador
Implementación de la base de datos	6 días	lunes, 17 de enero de 2011	miércoles, 19 de enero de 2011	21	Diseñador
Implementación del servidor	9 días	miércoles, 19 de enero de 2011	lunes, 31 de enero de 2011	22	Diseñador
Implementación de la aplicación móvil	30 días	miércoles, 19 de enero de 2011	miércoles, 16 de febrero de 2011	23CC	Diseñador
Fin de la fase de implementación	0 días	miércoles, 16 de febrero de 2011	miércoles, 16 de febrero de 2011	24	
Elaboración de documentación	30 días	miércoles, 16 de febrero de 2011	miércoles, 16 de marzo de 2011	24	R. de Documentación
Fin del proyecto	0 días	miércoles, 16 de marzo de 2011	miércoles, 16 de marzo de 2011	26	

Tabla 12: Planificación

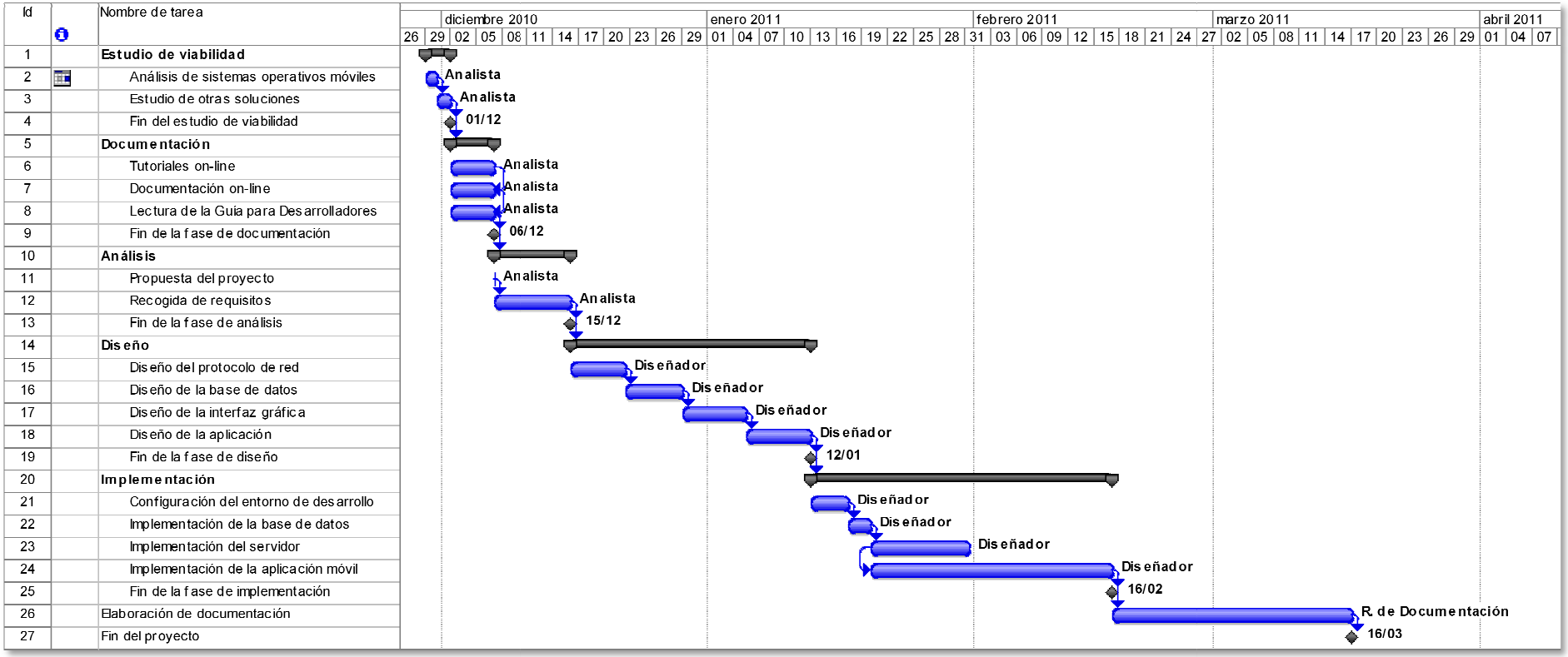


Ilustración 74: Planificación

8.2 Presupuesto

En esta sección se detalla el presupuesto del proyecto, identificando los costes en personal, de software y de hardware. La jornada laboral, tal y como se mencionó con anterioridad, es de 5 horas de lunes a miércoles.

8.2.1 Coste de personal

La siguiente tabla muestra el coste por hora de cada uno de los recursos de trabajo empleados en el proyecto:

Nombre	Número	Tasa estándar
Analista	1	30,00 €/hora
Diseñador	1	30,00 €/hora
Programador	1	15,00 €/hora
R. de Documentación	1	6,00 €/hora

Tabla 13: costes de personal

A partir de la planificación – para la que se ha usado el horario mencionado, y la tabla anterior, es sencillo obtener el coste de cada uno de los recursos de trabajo:

- Coste = 5.220 €

Tal y como muestra el siguiente diagrama:

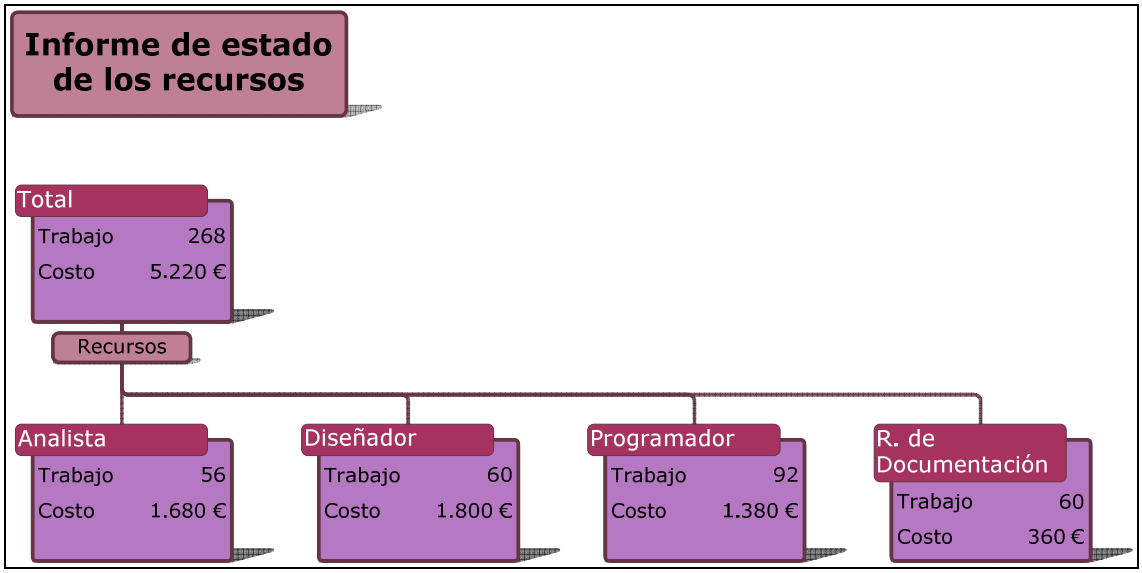


Ilustración 75: Informe de estado de los recursos

8.2.2 Coste de software y licencias

Para completar el proyecto ha sido necesario recurrir a diferentes aplicaciones y herramientas, que se enumeran a continuación junto con su coste imputable:

Nombre	Coste
Office 2007 Professional	708 €
Photoshop CS4 [6]	1.001 €
DynDNS Updater	0 €
SQLite Expert Personal 3	0 €
Eclipse + Android SDK	0 €
	1.709 €

Tabla 14: Costes de software

8.2.3 Coste de hardware

El coste de los componentes hardware necesario para desarrollar el proyecto se detalla a continuación.

Nombre	Coste imputable
Estación de trabajo	999 €
Teléfono HTC Desire HD	494 €
Teléfono HTC Hero	199 €
	1.692 €

Tabla 15: costes de hardware

8.2.4 Resumen de costes

El siguiente diagrama incluye el coste de todos los recursos empleados en el proyecto:

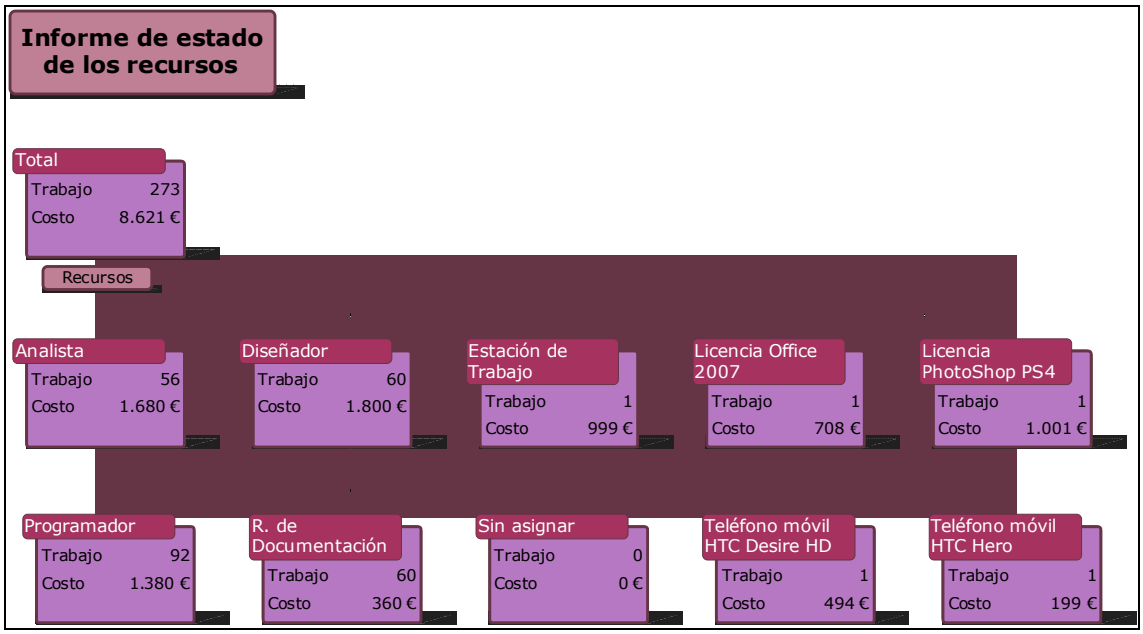


Ilustración 76: resumen de los costes de los recursos

Los costes del proyecto se reparten entre el material y los recursos de trabajo (analista, diseñador, programador y responsable de documentación). Siendo este úlitmo tipo de recurso el de mayor coste para el proyecto. Esta relación se aprecia en el siguiente gráfico:

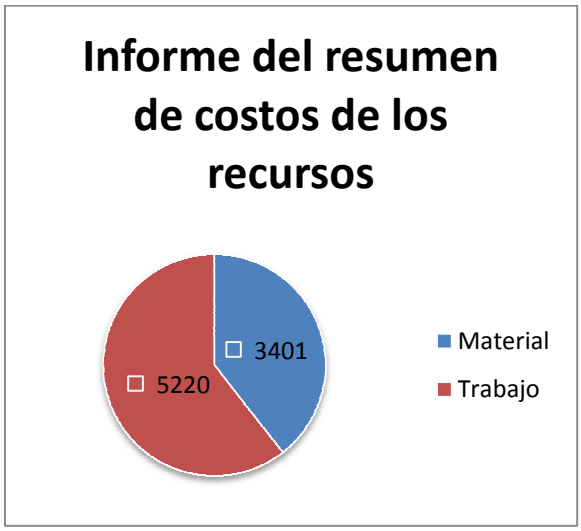


Ilustración 77: gráfico de costes de material y trabajo en el proyecto

8.2.5 Plantilla de presupuesto

A continuación se muestra el formulario de costes propuesto por la universidad, que incluye los costes detallados en las secciones anteriores. Los valores obtenidos son ligeramente diferentes, pues se tiene en cuenta la amortización del hardware, y suma un 20% de impuestos indirectos.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Miguel Degayón Cortés
2.- Departamento:
Informática
3.- Descripción del Proyecto:
- Titulo: **Diseño e implementación de un entrenador de cuestionarios para dispositivos Android**
- Duración (meses): **4 meses**
Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Miguel Degayón Cortés		Analista	0,43	3.937,50	1.680,00	
Miguel Degayón Cortés		Diseñador	0,46	3.937,50	1.800,00	
Miguel Degayón Cortés		Programador	0,70	1.968,75	1.380,00	
Miguel Degayón Cortés		R. de Documentación	0,46	787,50	360,00	
		Hombres mes	**2,041904762**	**Total**	5.220,00	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

Ilustración 78: primera parte de la plantilla de costes

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Teléfono HTC Desire HD	494,00	100	4	60	32,93
Teléfono HTC Hero	199,00	100	4	60	13,27
Equipo de trabajo	999,00	100	4	60	66,60
					0,00
					0,00
					0,00
				Total	112,80

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

Ilustración 79: segunda parte de la plantilla de costes

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
Licencia Microsoft Office Profesional		708,00
Licencia PhotoShop CS4		1.001,00
	Total	1.709,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,..

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	5.220
Amortización	113
Subcontratación de tareas	0
Costes de funcionamiento	1.709
Costes Indirectos	1.408
Total	8.450

Ilustración 80: última parte de la plantilla de coste

9 TRABAJOS FUTUROS Y CONCLUSIONES

Este capítulo versará sobre la opinión personal del autor sobre el proyecto además de comentar posibles futuros trabajos que mejoren y completen la aplicación desarrollada

9.1 Conclusiones

El desarrollo llegó a su fin, felizmente, y, con el proceso de desarrollo completado, es momento de hacer un repaso de estos meses dedicados al proyecto, a modo de conclusión.

Todo este tiempo me ha servido para comprender la plataforma Android: su arquitectura, la forma de construir una aplicación, basada en Actividades, Intents y services… y adquirir los conocimientos necesarios para desarrollar aplicaciones de cierta entidad. Desde que arrancó el proyecto hasta el momento en que escribo estas líneas, mi percepción del sistema operativo móvil ha cambiado sustancialmente: lo que en un principio no era más que un nombre en un catálogo de telefonía, se ha convertido en una plataforma con un sinfín de posibilidades para los desarrolladores de software (GPS, cámara de fotos y vídeo, conexión a internet, base de datos, telefonía…)

Fue una ventaja desde el principio contar con Java como lenguaje de programación – Aunque en Android no estén presentes toda sus librerías. Tras unos días de documentación, las primeras aplicaciones de prueba (un HelloWorld y un conversor de euros/pesetas), resueltas en poco tiempo, hacían pensar en un desarrollo fácil y sin demasiadas complicaciones… ¡Que equivocado estaba y cuanto faltaba por aprender! … Trabajar con los layouts y las diferentes vistas (muchas de las cuales tuve que aprender a personalizar), los ficheros de recurso en XML, el acceso a la base de datos, la conexión de red, la gestión de mensajes mediante Handlers, los selectores… largas tardes leyendo tutoriales y consultando (en ocasiones, sin obtener respuesta) foros. Sin embargo, ahora que el proyecto acabó, vuelve a mí, sin que sepa muy bien por qué, aquella sensación inicial, y todo me parece, de alguna forma, sencillo de nuevo.

Soy consciente (y si el lector tiene suficiente experiencia con este sistema operativo móvil, ya ha de saberlo) que, si bien ha sido mucho lo aprendido en este tiempo, queda aún mucho de Android (oportunidades y capacidades que no han sido explotadas) sobre lo que formarse. Sirva como ejemplo la posibilidad de usar el GPS o la cámara integrada, ¡o combinar ambas para hacer uso de la realidad aumentada!

Pero no todo ha sido nuevo en el desarrollo de TESTING, este proyecto ha supuesto también la oportunidad de aplicar muchos de lo aprendido en estos años (algunas ideas necesitaron de una puesta al día, otras, casi ausentes, cerca de ser olvidadas) de universidad. Me refiero a conocimientos relacionados con Bases de datos, el lenguaje XML, la arquitectura

cliente-servidor, patrones de diseño, UML , sockets… Este proyecto ha significado la ocasión perfecta para poner en práctica esas habilidades.

Hablaré, finalmente, de la aplicación en sí, del resultado de estos meses de trabajo. Y diré (y no pretendo ser orgulloso o engreído, pues en estos meses me he topado con proyectos geniales en la red, mejores que el mío) que estoy satisfecho con lo conseguido. Pasé mucho tiempo buscando aplicaciones similares (sólo hay que echarle un vistazo a los programas descargados en mi teléfono), investigando sobre posibles soluciones al problema planteado. Y creo que el nivel de TESTING, en más de un aspecto, supera a muchas de las aplicaciones similares de pago encontradas en el Market. Sé, sin embargo, que hay características mejorables que, de incluirse, harían de TESTING una aplicación más completa. Y es posible que lo haga. Que me permita el capricho de añadirle nuevas capacidades, de completar el servidor, o refinar la interfaz. Pero no ya no será necesario leer más de cien páginas sobre ello, paciente lector. Tendrás que buscarlo en el Market.

9.2 Mejoras futuras

Si bien estoy satisfecho con el resultado del desarrollo, no ignoro la existencia de funcionalidades que me habrían logrado mejorar lo conseguido tras el proyecto. Y, con él finalizado, es un buen momento para hacer repaso de estas características que mejorarían la aplicación y que por varios motivos () no se han incluido en la aplicación – en la mayoría de los casos por no aportar más conocimiento nada nuevo que aprender – ejemplo, para mejorar el feedback de los botones cambio color texto e imagen, tiene que ver con la personalización y el uso de selectores = para el fondo de la imagen. En otros, como el caso del servidor, porque se alejaba del objetivo proyecto – el estudio de la plataforma Android y el desarrollo de una aplicación móvil. A continuación resumen.

- **Servidor y aplicación web**

  En mi opinión, se trata de la mejora más evidente. Construir la aplicación web que permita al personal docente crear y/o subir los cuestionarios de los que se sirve la aplicación móvil.

- **Protocolo de red**

  En parte relacionado con el punto anterior (pues la aplicación web hace las veces de servidor en el protocolo). El protocolo de red también es susceptible de mejora. No existen intentos de reconexión, por poner un ejemplo.

- **Aplicación móvil**

  En cuanto a la aplicación móvil, existen varias funcionalidades que harían de TESTING un sistema más completo. Sería interesante que los usuarios pudieran crear sus propios cuestionarios, a parir de las preguntas de los tests descargados o creando nuevas preguntas desde cero. En relación a este punto, sería más que conveniente permitir editar, borrar y, tal vez compartir, estos tests creados por el usuario. La posibilidad de almacenar los resultados de los test realizados para conocer la evolución /mejora del alumno es, a mis ojos, más que interesante. Otras mejoras serían la inclusión de filtros, opciones de ordenación y búsqueda en la aplicación.

10 APÉNDICE I. Glosario

URL	Del inglés uniform resource locator. Localizador uniforme de recursos
NIA	Número de identificación de alumno
SMS	El servicio de mensajes cortos o SMS (Short Message Service)
SDK	Un software development kit (SDK) o kit de desarrollo de software es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores ....
smartphone	Un smartphone (teléfono inteligente en español) es un dispositivo electrónico que funciona como un teléfono móvil con características similares a las de un ordenador personal.
Acelerómetro	Es el instrumento que se utiliza para realizar las mediciones de vibraciones. Para obtener datos fiables de las mediciones, éstas se deben repetir varias veces y tener en cuenta los siguientes factores: Localización del punto de medida. Estimación de los niveles y tipos de vibración.
GUI	La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.
spinner	Un spinner es un elemento de las interfaces gráficas que permite al usuario ajustar un valor dentro de un cuadro de texto adjunto a dos flechas que apuntan en direcciones opuestas (generalmente una hacia arriba y otra hacia abajo). Generalmente puede saltarse hacia el siguiente valor haciendo clic en una de las flechas, o haciendo un clic sostenido para cambiar los valores más rápidamente.
feedback	Equivale a retroalimentación o retroacción, y consiste en introducir los resultados obtenidos como datos para

	considerar al inicio del nuevo proceso, lo que permitirá rectificar - si procede - dicho proceso.
tabView	Vista de Android compuesto por pestañas, de forma que únicamente es visible una de ellas.
trigger	Un trigger (o disparador) en una Base de datos , es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

11 APÉNDICE II. Referencias

[1] Apple
www.apple.com/es/

[2] DynDNS:
http://es.wikipedia.org/wiKI/DynDNS
http://www.dyndns.com/

[3] Guía de desarrollo para Android
http://developer.android.com/resources/articles/layout-tricks-reuse.html

[4] IOS
es.wikipedia.org/wiki/IOS

[5] Nokia
www.nokia.es

[6] PhotoShop
www.adobe.com/es/products/photoshop.html

[7] RIM
es.blackberry.com
www.rim.com

[8] Sqlite Expert
http://sqliteexpert.com/

[9] Symbian
es.wikipedia.org/wiki/Symbian_OS

[10] Windows Phone 7
www.microsoft.com/windowsphone/es-es/default.aspx